

The McGraw-Hill Companies



# **CRYPTOGRAPHY AND NETWORK SECURITY**

**ATUL KAHATE**



**SECOND EDITION**

Copyrighted material



**Tata McGraw-Hill**

Published by the Tata McGraw-Hill Publishing Company Limited,  
7 West Patel Nagar, New Delhi 110 008.

Copyright © 2003, 2008 by Tata McGraw-Hill Publishing Company Limited.

No part of this publication may be reproduced or distributed in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise or stored in a database or retrieval system without the prior written permission of the publishers. The program listings (if any) may be entered, stored and executed in a computer system, but they may not be reproduced for publication.

Fourth reprint 2008  
**RALYDRXRALLA**

This edition can be exported from India only by the publishers,  
Tata McGraw-Hill Publishing Company Limited.

ISBN-13: 978-0-07-064823-4  
ISBN-10: 0-07-064823-9

Managing Director: *Ajay Shukla*  
General Manager: Publishing—SEM & Tech Ed: *Vibha Mahajan*  
Asst. Sponsoring Editor: *Shalini Jha*  
Jr. Editorial Executive: *Nilanjan Chakravarty*  
Executive—Editorial Services: *Sohini Mukherjee*  
General Manager: Marketing—Higher Education & School: *Michael J. Cruz*  
Product Manager: SEM & Tech Ed: *Biju Ganesan*  
Controller—Production: *Rajender P Ghansela*  
Assistant General Manager—Production: *B L Dogra*  
Senior Production Executive: *Anjali Razdan*

Information contained in this work has been obtained by Tata McGraw-Hill, from sources believed to be reliable. However, neither Tata McGraw-Hill nor its authors guarantee the accuracy or completeness of any information published herein, and neither Tata McGraw-Hill nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that Tata McGraw-Hill and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.

Typeset at Tej Composers, WZ-391, Madipur Village, New Delhi 110 063 and printed at  
Anand Book Binding House, Delhi 110 031

Cover Printer: SDR Printers

**The McGraw-Hill Companies**

# Contents

<i>Foreword</i>	<i>xi</i>
<i>Note of Appreciation</i>	<i>xiii</i>
<i>Preface to the Second Edition</i>	<i>xv</i>
<i>Preface to the First Edition</i>	<i>xix</i>
<i>Important Terms and Abbreviations</i>	<i>xxi</i>
<b>1. Attacks on Computers and Computer Security</b>	<b>1</b>
1.1 Introduction	1
1.2 The Need for Security	1
1.3 Security Approaches	4
1.4 Principles of Security	7
1.5 Types of Attacks	12
Summary	33
Multiple-choice Questions	34
Exercises	36
Design/Programming Exercises	37
<b>2. Cryptography: Concepts and Techniques</b>	<b>38</b>
2.1 Introduction	38
2.2 Plain Text and Cipher Text	40
2.3 Substitution Techniques	41
2.4 Transposition Techniques	54
2.5 Encryption and Decryption	59
2.6 Symmetric and Asymmetric Key Cryptography	62
2.7 Steganography	73
2.8 Key Range and Key Size	74
2.9 Possible Types of Attacks	77
Summary	81
Multiple-choice Questions	83
Exercises	85
Design/Programming Exercises	85

<b>3. Symmetric Key Algorithms and AES</b>	<b>87</b>
3.1 Introduction	87
3.2 Algorithm Types and Modes	87
3.3 An Overview of Symmetric Key Cryptography	98
3.4 Data Encryption Standard (DES)	100
3.5 International Data Encryption Algorithm (IDEA)	115
3.6 RC4	123
3.7 RC5	125
3.8 Blowfish	131
3.9 Advanced Encryption Standard (AES)	137
<i>Summary</i>	148
<i>Multiple-choice Questions</i>	150
<i>Exercises</i>	152
<i>Design/Programming Exercises</i>	152
<b>4. Asymmetric Key Algorithms, Digital Signatures and RSA</b>	<b>153</b>
4.1 Introduction	153
4.2 Brief History of Asymmetric Key Cryptography	153
4.3 An Overview of Asymmetric Key Cryptography	154
4.4 The RSA Algorithm	156
4.5 Symmetric and Asymmetric Key Cryptography Together	160
4.6 Digital Signatures	165
4.7 Knapsack Algorithm	197
4.8 Some Other Algorithms	198
<i>Summary</i>	201
<i>Multiple-choice Questions</i>	201
<i>Exercises</i>	203
<i>Design/Programming Exercises</i>	203
<b>5. Digital Certificates and Public Key Infrastructure (PKI)</b>	<b>205</b>
5.1 Introduction	205
5.2 Digital Certificates	206
5.3 Private Key Management	237
5.4 The PKIX Model	239
5.5 Public Key Cryptography Standards (PKCS)	241
5.6 XML, PKI and Security	247
5.7 Creating Digital Certificates Using Java	252
<i>Summary</i>	260
<i>Multiple-choice Questions</i>	262
<i>Exercises</i>	263
<i>Design/Programming Exercises</i>	263

---

<b>6. Internet Security Protocols</b>	<b>265</b>
6.1 Introduction	265
6.2 Basic Concepts	265
6.3 Secure Socket Layer (SSL)	272
6.4 Transport Layer Security (TLS)	284
6.5 Secure Hyper Text Transfer Protocol (SHTTP)	284
6.6 Time Stamping Protocol (TSP)	285
6.7 Secure Electronic Transaction (SET)	286
<a href="#">6.8 SSL Versus SET</a>	<a href="#">298</a>
<a href="#">6.9 3-D Secure Protocol</a>	<a href="#">299</a>
6.10 Electronic Money	302
<a href="#">6.11 Email Security</a>	<a href="#">307</a>
6.12 Wireless Application Protocol (WAP) Security	327
6.13 Security in GSM	330
6.14 Security in 3G	332
<i>Summary</i>	335
<i>Multiple-choice Questions</i>	337
<a href="#">Exercises</a>	<a href="#">338</a>
<a href="#">Design/Programming Exercises</a>	<a href="#">339</a>
<b>7. User Authentication and Kerberos</b>	<b>340</b>
<a href="#">7.1 Introduction</a>	<a href="#">340</a>
<a href="#">7.2 Authentication Basics</a>	<a href="#">340</a>
7.3 Passwords	341
<a href="#">7.4 Authentication Tokens</a>	<a href="#">354</a>
7.5 Certificate-based Authentication	365
<a href="#">7.6 Biometric Authentication</a>	<a href="#">371</a>
7.7 Kerberos	372
7.8 Key Distribution Center (KDC)	378
7.9 Security Handshake Pitfalls	379
7.10 Single Sign On (SSO) Approaches	387
<i>Summary</i>	388
<i>Multiple-choice Questions</i>	390
<i>Exercises</i>	391
<a href="#">Design/Programming Exercises</a>	<a href="#">391</a>
<b>8. Cryptography in Java, .NET and Operating Systems</b>	<b>393</b>
8.1 Introduction	393
8.2 Cryptographic Solutions Using Java	393
<a href="#">8.3 Cryptographic Solutions Using Microsoft .NET Framework</a>	<a href="#">400</a>
8.4 Cryptographic Toolkits	403

8.5 Security and Operating Systems	404
8.6 Database Security	409
<i>Summary</i>	426
<i>Multiple-choice Questions</i>	427
<i>Exercises</i>	428
<i>Design/Programming Exercises</i>	428
<b>9. Network Security, Firewalls and Virtual Private Networks (VPN)</b>	<b>430</b>
9.1 Introduction	430
9.2 Brief Introduction to TCP/IP	430
<u>9.3 Firewalls</u>	<u>435</u>
<u>9.4 IP Security</u>	<u>452</u>
9.5 Virtual Private Networks (VPN)	469
<u>9.6 Intrusion</u>	<u>472</u>
<i>Summary</i>	476
<i>Multiple-choice Questions</i>	478
<i>Exercises</i>	479
<i>Design/Programming Exercises</i>	480
<b>10. Case Studies on Cryptography and Security</b>	<b>481</b>
<u>10.1 Introduction</u>	<u>481</u>
<u>10.2 Cryptographic Solutions—A Case Study</u>	<u>481</u>
10.3 Single Sign On (SSO)	488
10.4 Secure Inter-branch Payment Transactions	491
<u>10.5 Denial Of Service (DOS) Attacks</u>	<u>496</u>
<u>10.6 IP Spoofing Attacks</u>	<u>498</u>
<u>10.7 Cross Site Scripting Vulnerability (CSSV)</u>	<u>499</u>
<u>10.8 Contract Signing</u>	<u>501</u>
<u>10.9 Secret Splitting</u>	<u>501</u>
<u>10.10 Virtual Elections</u>	<u>502</u>
10.11 Secure Multiparty Calculation	504
<u>10.12 Creating a VPN</u>	<u>505</u>
10.13 Cookies and Privacy	506
<i>Appendix A: Mathematical Background</i>	<i>507</i>
<i>Appendix B: Number Systems</i>	<i>516</i>
<i>Appendix C: Information Theory</i>	<i>521</i>
<i>Appendix D: Real-life Tools</i>	<i>523</i>
<i>Appendix E: Web Resources</i>	<i>524</i>
<i>Appendix F: A Brief Introduction to ASN, BER, DER</i>	<i>527</i>
<i>References</i>	<i>533</i>
<i>Index</i>	<i>535</i>



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

DES was based on symmetric key cryptography. In this method, the same key was used for encryption and decryption. Therefore, both the sender and the receiver had to know and agree about the key in advance. This posed a serious problem in the Internet world where many users were expected to send/receive messages to/from a server in a secure fashion. How should the keys for each pair be determined, exchanged and kept a secret?

RSA solved this problem by designing a key pair: one for encryption and the other for decryption. In fact, if a message is encrypted by one key, only the other key in a pair can decrypt it. The concept was based on the idea that it is virtually impossible to factor the product of two very large prime numbers (e.g., with 100 or more digits each). In RSA, two very large prime numbers can be used as a key pair: one as a public key and the other one as a private key. In fact, the entire security infrastructure was built on the Public Key Cryptography (PKC) concept and was called Public Key Infrastructure (PKI).

RSA was designed by Rivest, Shamir and Adleman. They posed a challenge and declared a reward of \$100 for anyone who would decrypt the message, which they had encrypted. This was published in 1997 by Martin Gardner in *Scientific American* in a widely read column, *Mathematical Games*, which he edited. It was estimated that even using the most powerful computers available at that time, it would take about 40 quadrillion years to decrypt this message!! Later in 1978, RSA formally introduced the PKC System.

With the growth of the Internet, the need for secure data transmission increased manyfold. In fact, it became a pre-condition to the usage of the Internet for business transactions. In 2002 alone, security related frauds cost businesses about 6 billion US dollars. Security, therefore, is a major concern in the Internet world—especially for financial transactions.

It is on this backdrop that the current text is extremely important. It is virtually impossible to architect a web-based software system without taking into account the security concerns. This book is, therefore, quite timely. There are a few books available in the market on the same topic, but, in my opinion, the current text stands out due to its simplicity. The book can be understood by virtually anybody who has some fundamental understanding of computing environments. The language is lucid and the book has a number of diagrams to enhance the comprehension. Interestingly, despite its simplicity, it does not lose its rigor and depth. Therefore, I feel that this book will be of tremendous value not only to the students who can use it as a text but also for software managers and software architects who can use it as a reference guide.

I have been lucky enough to have co-authored a book called *Web Technologies* with Atul. I think he is one of the most intelligent, systematic and thorough individuals whom I have met in my career. Despite his technical excellence, he is quite unassuming which I value the most. I have observed that he has many other interests than just technology, music and cricket being some of them. He is regarded as an authentic cricket statistician. And above all, I find him extremely humane which I think, makes him one of the finest human beings in the ultimate analysis.

It has been a great pleasure knowing him and working with him. I wish him all the very best in his future endeavours.

**ACHYUT S GODBOLE**

*CEO—Apar Technologies, Mumbai*

*(Author of Operating Systems, Data Communications and Networks,  
and Web Technologies, all published by Tata McGraw-Hill)*



# Note of Appreciation

Information security has become a very critical aspect of modern computing systems. With the global acceptance of the Internet, virtually every computer in the world today is connected to every other. While this has created tremendous productivity and unprecedented opportunities in the world we live in, it has also created new risks for the users of these computers. The users, businesses and organisations worldwide have to live with a constant threat from hackers and attackers, who use a variety of techniques and tools in order to break into computer systems, steal information, change data and cause havoc.

It is in this context that Atul Kahate came up with his second book: *Cryptography and Network Security*. Atul's knowledge is not only based on serious study and research but also in hands-on solving of the challenges encountered in real-life situations, dealing with Public Key Infrastructure (PKI) and related areas for building and testing cryptography and security in complex software systems. The i-flex Centre of Excellence (COE) for Payment Systems, Pune, India, has given Atul a suitable environment for his study and research for most of the last six years.

Acquiring and developing knowledge is a great achievement. Sharing it in a fashion that would help all others understand it is even better. It is a very tedious, time-consuming and difficult task to explain something that you know, in a useful manner to others. Atul has achieved the same by putting his knowledge in a logical and practical manner, resulting in this comprehensive book on the subject. Atul had earlier co-authored a book on Web Technologies (also published by Tata McGraw-Hill), which attracted very good feedback.

We, at i-flex, are extremely happy and proud of Atul's achievement of completing this book as well as many others on computer technology. I am quite sure that he would author many more books in the future. Apart from the depth of coverage, broad areas of topics discussed, the two features that enable this book to stand out are the lucid language and the illustrative approach. The step-by-step fashion in which Atul has dealt with the complex topics of Cryptography and Security would no doubt help the reader understand all the key concepts with ease. I am quite sure that the book would be of a great help to all students, teachers and IT professionals at various levels.

This also goes on to show that one must not restrict one's activities to mundane tasks that we usually end up carrying out on any working day. If we put a little more effort and have higher aims, we all can excel in different aspects of our profession (and even outside of our professional boundaries).

On behalf of all of us at i-flex solutions, I wish Atul a grand success with this book.

**RAJESH HUKKU**  
*Chairman*  
*i-flex solutions limited*

## Preface to the Second Edition

Having worked in the area of Information Technology for about six years (in 2001), I had read a lot about information security, and how to achieve it. However, my concepts were vague, and I knew the technology of security in bits and pieces. This was quite annoying, as it never gave a feeling of satisfaction. It was as if I did not know the complete picture. For example, I did know that number systems played an important role in cryptography, but did not know how much I should know about them to understand the concepts thoroughly. Similarly, I knew that digital certificates and Public Key Infrastructure (PKI) were quite wonderful technologies, but knew only to some extent as to how they worked. Numerous other examples can be given.

Then I got an opportunity to lead an information security project in i-flex solutions limited. I knew that I could learn a lot simply by working on that project. However, I also felt very strongly that until I was thorough with all the aspects of computer security/cryptography myself, I would not be able to do true justice to this project. It was for this reason that I took up the task of studying each and every aspect of these technologies. Unfortunately, there were a lot of hurdles. The main hurdle was that there was not a single book that explained all that I wanted, and more importantly, in the manner that I wanted. My colleagues in the project also expressed this feeling on many occasions. The information available was scattered, quite complex to understand, and not explained to the level that makes one completely understand what is going on.

The struggle for learning was quite wonderful! However, it also convinced me that I should make an attempt to explain what I know, in a very simple manner, so that others who venture into this area do not have to struggle the way I did. This is perhaps the main intention behind this book. In simple terms, it is something, which makes me feel, 'if only such a book were available when I started exploring and learning about security/cryptography'. The biggest satisfaction will be if and when readers in similar situations, feel contented after reading this book.

The first edition of this book was published in early 2003. At that time, there were very few books on the subject, and the ones that existed were quite complex to comprehend. Hence, I had made a genuine attempt to simplify the subject to the maximum possible extent. I had not written the book with any specific aims of addressing the needs of some syllabi. I had written it in a manner that I felt made understanding the subject very easy, more than anything else. To my surprise, in the last four and half years, not only has the book been used in almost all syllabi in India and many other countries, but in addition, several syllabi/courses have been designed around this book. This has reinforced my belief that the sequencing and structuring of the contents of the book is largely correct. This belief has been graciously greeted by the readers of the first edition of the book, which has seen 8 reprints, an international edition, and a Chinese translation, too!



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

- Interruption** Attacker creating a situation where the availability of a system is in danger. Same as *Masquerade*.
- IP Security (IPSec)** Protocol to encrypt messages at the network layer.
- Issuer** Bank/financial institution that facilitates a cardholder to make credit card payments on the Internet.
- Java applet** Small client-side program that gets downloaded along with a Web page, and executes inside the browser. This is a Sun technology. Java applets are somewhat similar to ActiveX controls.
- Java Cryptography Architecture (JCA)** Java's cryptography mechanism, in the form of APIs.
- Java Cryptography Extensions (JCE)** Java's cryptography mechanism, in the form of APIs.
- Kerberos** Single Sign On (SSO) mechanism, that allows a user to have a single user id and password to access multiple resources/systems.
- Key** The secret information in a cryptographic operation.
- Key Distribution Center (KDC)** A central *authority* dealing with keys for individual computers (nodes) in a computer network.
- Key wrapping** See *Digital envelope*.
- Known plaintext attack** In this case, the attacker knows about some pairs of plain text and corresponding cipher text for those pairs. Using this information, the attacker tries to find other pairs, and therefore, know more and more of the plain text.
- Lightweight Directory Access Protocol (LDAP)** Protocol that allows easy storage and retrieval of information at/from a central place.
- Linear cryptanalysis** An attack based on linear approximations.
- Lucifer** One symmetric key encryption algorithm.
- Man-in-the-middle attack** A form of attack in which the attacker intercepts the communication between two parties, and fools them to believe that they are communicating with each other, whereas they actually communicate with the attacker. Same as *bucket brigade attack*.
- Masquerade** Attacker creating a situation where the availability of a system is in danger. Same as *Interruption*.
- MD5** Message digest algorithm, now seems vulnerable to attacks.
- Merchant** Person/organization, who sets up an online shopping site, and accepts electronic payments.
- Message Authentication Code (MAC)** See *HMAC*.
- Message digest** *Finger print* of a message, same as *Hash*. Identifies a message uniquely.
- Microsoft Cryptography Application Programming Interface (MS-CAPI)** Microsoft's cryptography mechanism, in the form of APIs.
- Modification** Attack on a message where its contents are changed.
- Mono-alphabetic Cipher** Technique of encryption in which one plain text character is replaced with one cipher text character, at a time.
- Multi-factor authentication** Authentication mechanism, which involves the party to be authenticated concerned with multiple factors (e.g. *know* something, *be* something and *have* something).
- Mutual authentication** In mutual authentication, A and B both authenticate each other.
- Network level attack** Security attacks attempted at the network/hardware level.
- Non-repudiation** Provision whereby the sender of a message cannot refuse having sent it, later on, in the case of a dispute.
- One-Time Pad** Considered very secure, this method involves the usage of a key, which is used only once and then discarded forever.
- One-time password** Technology that authenticates user based on passwords that are generated dynamically, used once, and then destroyed.
- One-way authentication** In this scheme, if there are two users A and B, B authenticates A, but A does not authenticate B.

- Online Certificate Status Protocol (OCSP)** Online protocol to check the status of a digital certificate.
- Output Feedback (OFB)** Mode of chaining.
- Packet filter** Firewall that filters individual packets based on rules. Works at the network layer.
- Passive attack** Form of attack on security where the attacker does not make an attempt to change the contents of the message.
- Password** Authentication mechanism that requires a user to enter a secret piece of information (i.e. the password) when challenged.
- Password policy** Statement outlining the structure, rules and mechanisms of passwords, in an organization.
- Pharming** Modifying the Domain Name System (DNS) so as to direct genuine URLs to false IP addresses of attackers.
- Phishing** Technique used by attackers to fool innocent users into providing confidential/personal information.
- Plain text** Message in an understandable/readable form, same as *Clear text*.
- Playfair Cipher** A cryptographic technique that is used for manual encryption of data. This scheme was invented by Charles Wheatstone in 1854.
- Polygram Substitution Cipher** Technique of encryption where one block of plain text is replaced with another, at a time.
- Pretty Good Privacy (PGP)** Protocol for secure email communications, developed by Phil Zimmerman.
- Privacy Enhanced Mail (PEM)** Protocol for secure email communications, developed by Internet Architecture Board (IAB).
- Proof Of Possession (POP)** Establishing the proof that a user possesses the private key corresponding to the public key, as specified in the user's digital certificate.
- Proxy server** Type of firewall that filters packets at the application layer of TCP/IP stack. Same as *Application gateway* or *Bastion host*.
- Pseudocollision** Specific case of collision in the MD5 algorithm.
- Psuedo-random number** Random number generated using computers.
- Public Key Cryptography Standards (PKCS)** Standards developed by RSA Security Inc for the Public Key Infrastructure (PKI) technology.
- Public Key Infrastructure (PKI)** Technology for implementing ansymmetric key cryptography, with the help of message digests, digital signatures, encryption and digital certificates.
- Public Key Infrastructure X.509 (PKIX)** Model to implement PKI.
- Rail Fence Technique** Example of transposition technique.
- RC5** Symmetric key block encryption algorithm, involving variable length keys.
- Reference monitor** Central entity, which is responsible for all the decisions related to access control of computer systems.
- Registration Authority (RA)** Agency that takes some of the jobs of a Certification Authority (CA) on itself, and helps the CA in many ways.
- Replay attack** Form of attack wherein an attacker gets hold of a legal message, and attempts a re-transmission of the same at a later point of time.
- Replay attack** Attack on a system wherein the attacker gets hold of a message, and attempts to re-send it, hoping that the receiver does not detect this as a message sent twice.
- Roaming certificate** Digital certificate, which can be carried along as users move from one computer/location to another.
- RSA algorithm** Asymmetric key algorithm, widely used for encryption and digital signatures.
- Running Key Cipher** Techique where some portion of text from a book is used as the key.
- Secure Electronic Transaction (SET)** Protocol developed jointly by MasterCard, Visa and many other companies for secure credit card payments on the Internet.

**Secure MIME (S/MIME)** Protocol that adds security to the basic Multipurpose Internet Mail Extensions (MIME) protocol.

**Secure Socket Layer (SSL)** Protocol developed by Netscape communications for secure exchange of information between a Web browser and a Web server over the Internet.

**Self-signed certificate** Digital certificate, wherein the subject name and the issuer name are the same, and is signed by the issuer (which is also the subject). Usually the case only with CA certificates.

**SHA** Message digest algorithm, now preferred as the standard algorithm of choice.

**Signed Java applet** Technology to make Java applets more trustworthy.

**Simple Certificate Validation protocol (SCVP)** Enhancement of the basic Online Certificate Status Protocol (OCSP). Allows checks other than only the status of the certificate, unlike OCSP.

**Simple Columnar Transposition Technique** Variation of the basic transposition technique such as Rail Fence Technique.

**Simple Columnar Transposition Technique with multiple rounds** Variation of Simple Columnar Transposition Technique.

**Single Sign On (SSO)** Technology providing the users a single user id and password to access multiple systems/applications.

**Stream cipher** Technique encrypting one bit at a time.

**Substitution Cipher** Cryptographic technique involving the replacement of plain text characters with other characters.

**Symmetric Key Cryptography** Cryptographic technique where the same key is used for encryption and decryption operations.

**Time Stamping Authority (TSA)** Notary-like authority, which can vouch for the availability/creation of a digital document at a particular point of time.

**Time Stamping Protocol (TSP)** Protocol using which a Time Stamping Authority (TSP) vouches for the availability/creation of a digital document at a particular point of time.

**Time-based token** Type of authentication token.

**Traffic analysis** Mechanism whereby an attacker examines the packets moving across a network, and uses this information to launch an attack.

**Transport Layer Security (TLS)** Protocol similar to SSL.

**Transposition Cipher** Cryptographic technique involving the re-arrangement of plain text characters in some other form.

**Triple DES** Modified version of DES, involves 128-bit or 168-bit keys.

**Trojan horse** Small program that does not attempt to delete anything on the user's disk, but instead, replicates itself on the computer/networks.

**Trusted system** Computer system that can be trusted to a certain extent in terms of implementing the designated security policy.

**Vernam Cipher** See *One-time pad*.

**Virtual Private Network (VPN)** Technology that makes use of the existing Internet as a private network, using cryptographic techniques.

**Virus** Small program that causes harm to a user's computer and performs destructive activities.

**Wireless Transport Layer Security (WTLS)** Layer in WAP for facilitating secure communications between a client and a server.

**Worm** Small program, which does not damage a computer/network, but consumes resources, slowing it down considerably.

**X.500** Standard name for the LDAP technology.

**X.509** Format for digital certificate contents and structure.

**XML digital signatures** Technology that allows signing of specific portions of a message.

# Attacks on Computers and Computer Security



## 1.1 Introduction

This is a book on network and Internet security. Before we understand the various concepts and technical issues related to security (i.e. trying to understand *how* to protect), it is essential to know *what* we are trying to protect. The various dangers when we use computers, computer networks and the biggest network of them all, the Internet and the likely pitfalls. The consequences of not setting up the right security policies, framework and technology implementations. This chapter attempts to clarify these basic concepts.

We start with a discussion of the basic question: Why is security required in the first place? People sometimes say that security is like statistics: the extent of data it reveals is trivial, the extent of data it conceals is vital! In other words, the right security infrastructure opens up just enough doors that are mandatory. It protects everything else. We discuss a few real-life incidents that should prove beyond doubt that security cannot simply be compromised. Especially these days when serious business and other types of transactions are being conducted over the Internet to such a large extent, inadequate or improper security mechanisms can bring the whole business down or play havoc with people's lives!

We then discuss the key principles of security. These principles help us identify the various areas, which are crucial while determining the security threats and possible solutions to tackle them. Since electronic documents and messages are now becoming equivalent to paper documents in terms of their legal validity and binding, we examine the various implications in this regard.

This is followed by a discussion of the types of attacks. There are certain theoretical concepts associated with attacks and there is a practical side to it as well. We shall discuss all these aspects.

Finally, we discuss some modern security problems. This will pave the way for further discussions of network and Internet security concepts.



## 1.2 The Need for Security

### 1.2.1 Basic Concepts

Most initial computer applications had *no* or at best, *very little* security. This continued for a number of years until the importance of data was truly realized. Until then, computer data was considered to be



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



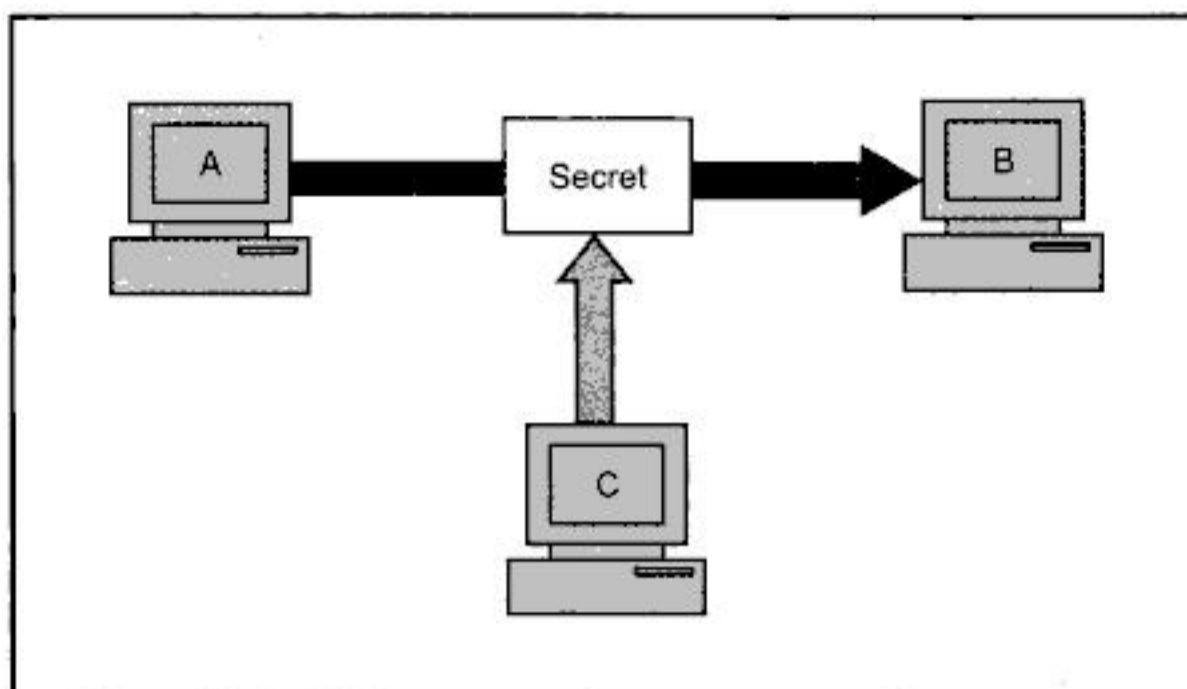
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



┆ Fig. 1.4 *Loss of confidentiality*

shall use the term A to mean the *user A*, B to mean *user B*, etc. although we shall just show the computers of user A, B, etc.). Another user C gets access to this message, which is not desired and therefore, defeats the purpose of confidentiality. Example of this could be a confidential email message sent by A to B, which is accessed by C without the permission or knowledge of A and B. This type of attack is called as **interception**.

*Interception causes loss of message confidentiality.*

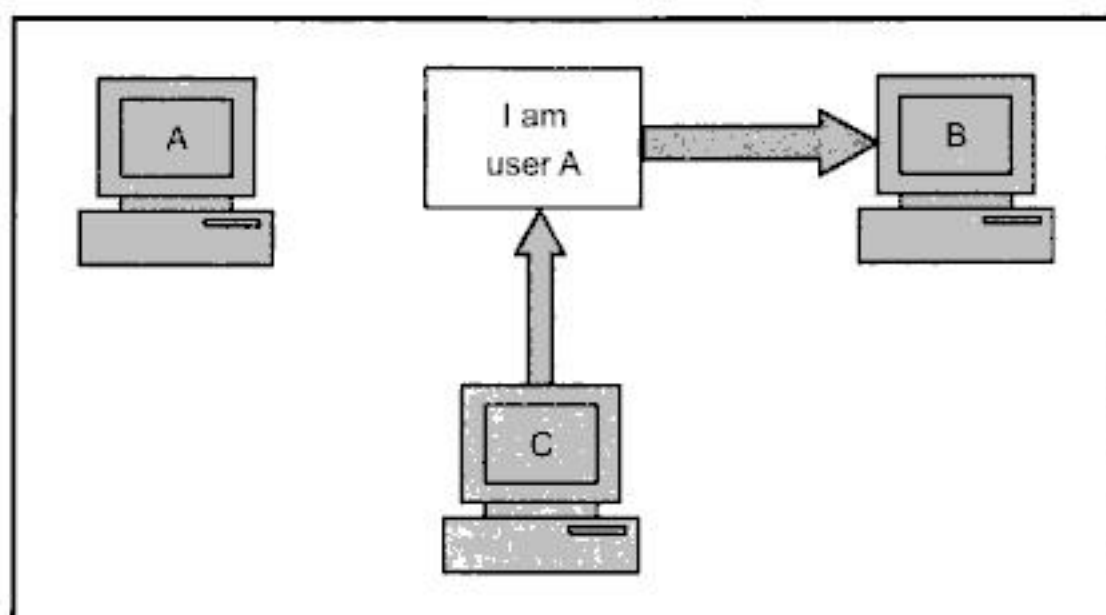
### 1.4.2 Authentication

*Authentication* mechanisms help establish proof of identities. The authentication process ensures that the origin of a electronic message or document is correctly identified. For instance, suppose that user C sends an electronic document over the Internet to user B. However, the trouble is that user C had posed as user A when she sent this document to user B. How would user B know that the message has come from user C, who is posing as user A? A real life example of this could be the case of a user C, posing as user A, sending a funds transfer request (from A's account to C's account) to bank B. The bank might happily transfer the funds from A's account to C's account – after all, it would think that user A has requested for the funds transfer! This concept is shown in Fig. 1.5. This type of attack is called as **fabrication**.

*Fabrication is possible in absence of proper authentication mechanisms.*

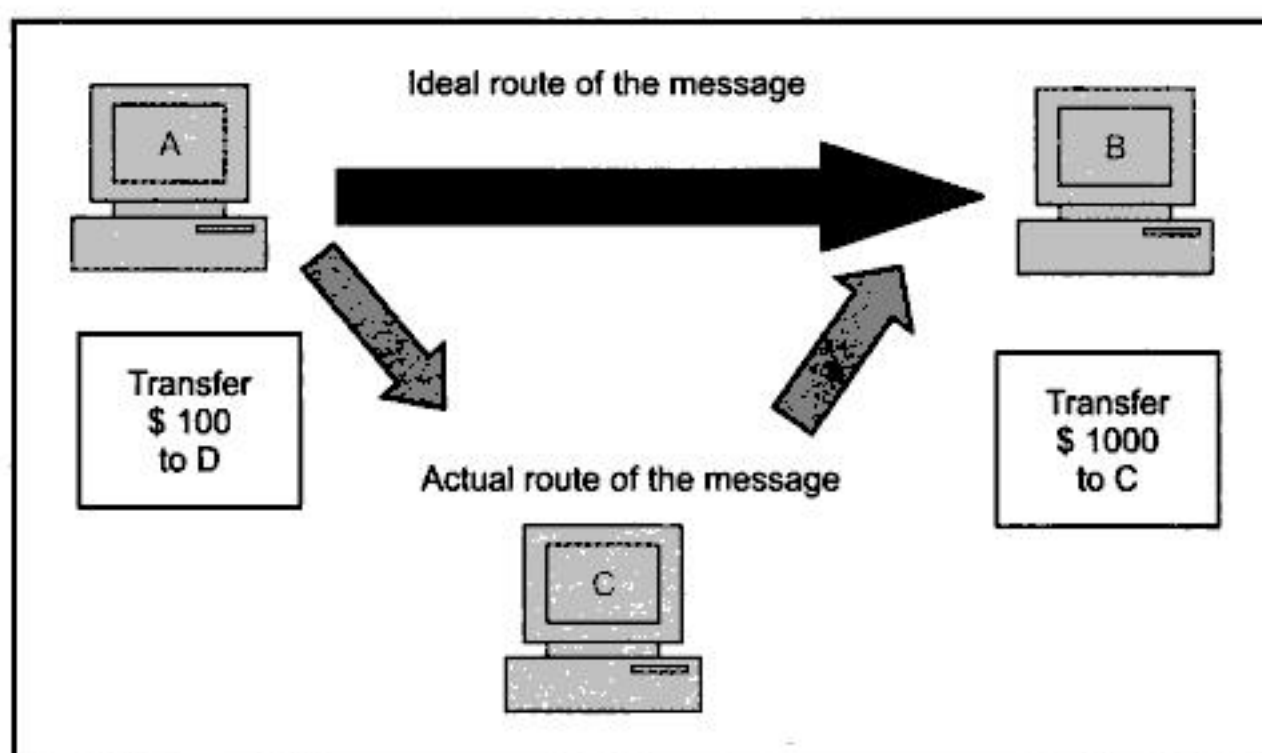
### 1.4.3 Integrity

When the contents of a message are changed after the sender sends it, but before it reaches the intended recipient, we say that the *integrity* of the message is lost. For example, suppose you write a check for \$100 to pay for the goods bought from the US. However, when you see your next account statement, you are startled to see that the check resulted in a payment of \$1000! This is the case for loss of message integrity. Conceptually, this is shown in Fig. 1.6. Here, user C tampers with a message originally sent by user A, which is actually destined for user B. User C somehow manages to access it, change its contents



┆ Fig. 1.5 *Absence of authentication*

and send the changed message to user B. User B has no way of knowing that the contents of the message were changed after user A had sent it. User A also does not know about this change. This type of attack is called as **modification**.



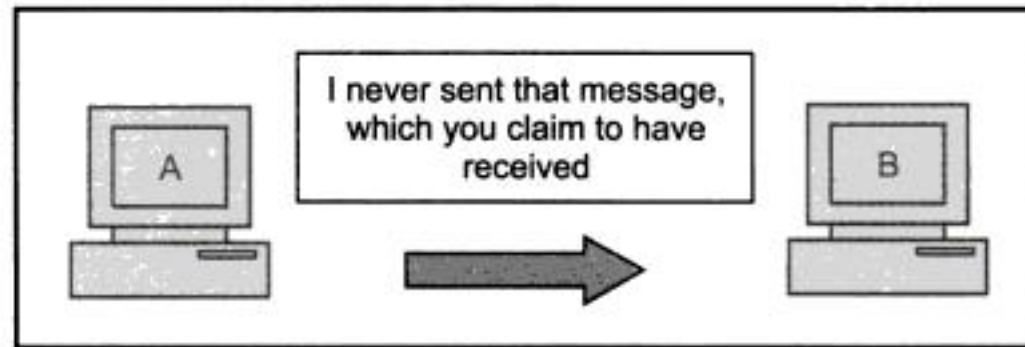
┆ Fig. 1.6 *Loss of integrity*

*Modification causes loss of message integrity.*

#### 1.4.4 Non-repudiation

There are situations where a user sends a message and later on refuses that she had sent that message. For instance, user A could send a funds transfer request to bank B over the Internet. After the bank performs the funds transfer as per A's instructions, A could claim that she never sent the funds transfer instruction to the bank! Thus, A repudiates or denies, her funds transfer instruction. The principle of *non-repudiation* defeats such possibilities of denying something, having done it. This is shown in Fig. 1.7.

*Non-repudiation does not allow the sender of a message to refute the claim of not sending that message.*



┆ Fig. 1.7 *Establishing non-repudiation*

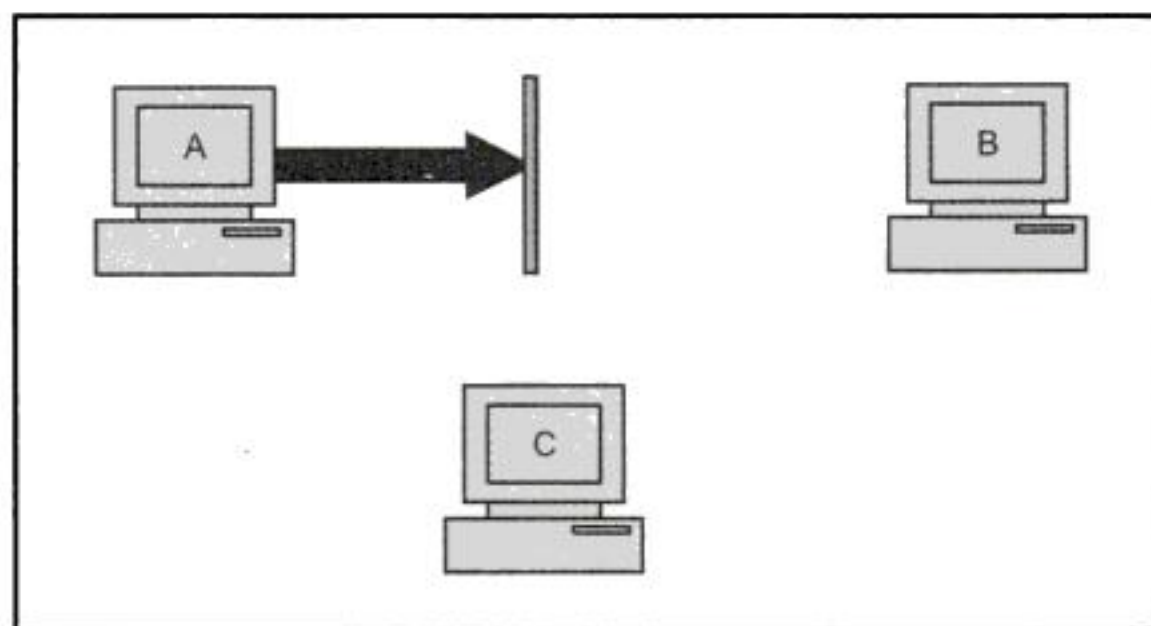
### 1.4.5 Access Control

The principle of *access control* determines *who* should be able to access *what*. For instance, we should be able to specify that user A can view the records in a database, but cannot update them. However, user B might be allowed to make updates as well. An access control mechanism can be set up to ensure this. Access control is broadly related to two areas: *role management* and *rule management*. Role management concentrates on the user side (which user can do what), whereas rule management focuses on the resources side (which resource is accessible and under what circumstances). Based on the decisions taken here, an access control matrix is prepared, which lists the users against a list of items they can access (e.g. it can say that user A can write to file X, but can only update files Y and Z). An **Access Control List (ACL)** is a subset of an access control matrix.

*Access control specifies and controls who can access what.*

### 1.4.6 Availability

The principle of *availability* states that resources (i.e. information) should be available to authorized parties at all times. For example, due to the intentional actions of an unauthorized user C, an authorized user A may not be able to contact a server computer B, as shown in Fig. 1.8. This would defeat the principle of availability. Such an attack is called as **interruption**.



┆ Fig. 1.8 *Attack on availability*

*Interruption puts the availability of resources in danger.*

We may be aware of the traditional OSI standard for Network Model (titled OSI Network Model 7498-1), which describes the seven layers of the networking technology (application, presentation, session, transport, network, data link and physical). A very less known standard on similar lines is the **OSI standard for Security Model** (titled OSI Security Model 7498-2). This also defines seven layers of security in the form of:

- Authentication
- Access control
- Non repudiation
- Data integrity
- Confidentiality
- Assurance or Availability
- Notarization or Signature

We shall be discussing upon most of these topics in this book.

Having explained the various principles of security, let us now discuss the various types of attacks that are possible, from a technical perspective.

#### 1.4.7 Ethical and Legal Issues

Many ethical and legal issues in computer security systems seem to be in the area of the individual's right to privacy versus the greater good of a larger entity (e.g. a company, society, etc.) For example, tracking how employees use computers, crowd surveillance, managing customer profiles, tracking a person's travel with a passport, location tracking so as to spam cell phone with text message advertisements and so on. A key concept in resolving this issue is to find out is a person's expectation of privacy.

Classically, the ethical issues in security systems are classified into the following four categories:

- Privacy – This deals with the right of an individual to control personal information.
- Accuracy – This talks about the responsibility for the authenticity, fidelity and accuracy of information.
- Property – Here we find out the owner of the information. We also talk about who controls access.
- Accessibility – This deals with the issue of the type of information an organization has the right to collect. And in that situation, it also expects to know the measures which will safeguard against any unforeseen eventualities.

Privacy is the protection of personal or sensitive information. Individual privacy is the desire to be *left alone* as an extension of our *personal space* and may or may not be supported by local regulations or laws. Privacy is subjective. Different people have different ideas of what privacy is and how much privacy they will trade for safety or convenience.

When dealing with legal issues, we need to remember that there is a hierarchy of regulatory bodies that govern the legality of information security. We can roughly classify them as follows.

- International: e.g. International Cybercrime Treaty
- Federal: e.g. FERPA, GLB, HIPAA, DMCA, Teach Act, Patriot Act, Sarbanes-Oxley Act, etc.
- State: e.g. UCITA, SB 1386, etc.
- Organization: e.g. Computer use policy



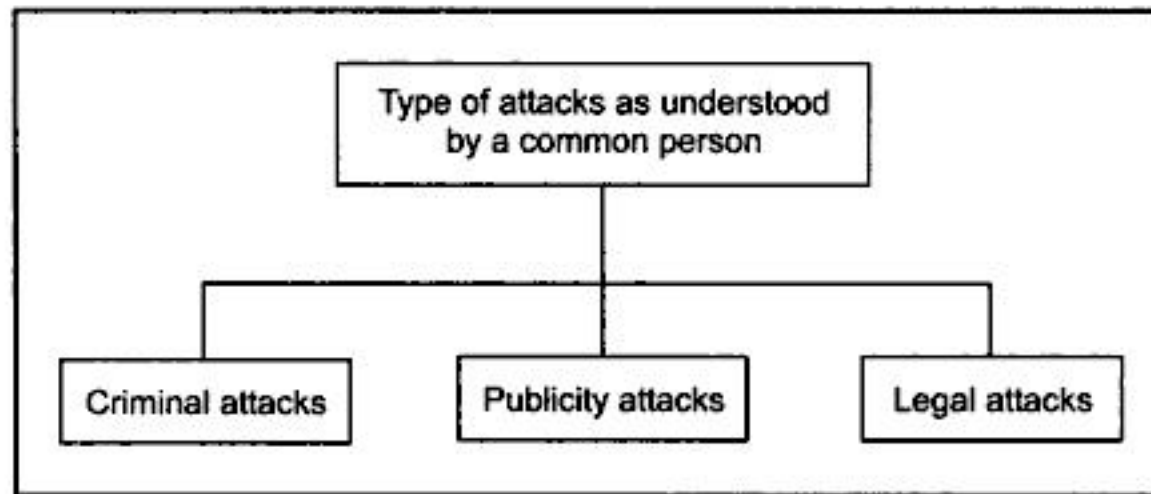


## 1.5 Types of Attacks

We shall classify attacks with respect to two views: the common person's view and a technologist's view.

### 1.5.1 Attacks: A General View

From a common person's point of view, we can classify attacks into three categories, as shown in Fig. 1.9.



┆ Fig. 1.9 Classification of attacks as understood in general terms

Let us now discuss these attacks.

**Criminal Attacks** Criminal attacks are the simplest to understand. Here, the sole aim of the attackers is to maximize financial gain by attacking computer systems. Table 1.1 lists some forms of criminal attacks.

**Publicity Attacks** Publicity attacks occur because the attackers want to see their names appear on television news channels and newspapers. History suggests that these types of attackers are usually not hardcore criminals. They are people such as students in universities or employees in large organizations, who seek publicity by adopting a novel approach of attacking computer systems.

One form of publicity attacks is to damage (or deface) the Web pages of a site by attacking it. One of the most famous such attacks occurred on the *US Department of Justice's* Web site in 1996. The *New York Times* home page was also famously defaced two years later.

**Legal Attacks** This form of attack is quite novel and unique. Here, the attacker tries to make the judge or the jury doubtful about the security of a computer system. This works as follows. The attacker attacks the computer system and the attacked party (say a bank or an organization) manages to take the attacker to the court. While the case is being fought, the attacker tries to convince the judge and the jury that there is inherent weakness in the computer system and that she has done nothing wrongful. The aim of the attacker is to exploit the weakness of the judge and the jury in technology matters.

For example, an attacker may sue a bank for performing an online transaction, which she never wanted to perform. In court, she could innocently say something like *The bank's Web site asked me to enter a password and that is all that I provided; I do not know what happened thereafter.* A judge is likely to sympathize with the attacker!

Table 1.1 Types of Criminal Attacks

Attack	Description
<b>Fraud</b>	Modern fraud attacks concentrate on manipulating some aspects of electronic currency, credit cards, electronic stock certificates, checks, letters of credit, purchase orders, ATMs, etc.
<b>Scams</b>	Scams come in various forms, some of the most common ones being sale of services, auctions, multi-level marketing schemes, general merchandise and business opportunities, etc. People are enticed to send money in return of great profits, but end up losing their money. A very common example is the <i>Nigeria scam</i> , where an email from Nigeria (and other African countries) entices people to deposit money into a bank account with a promise of hefty gains. Whosoever gets caught in this scam loses money heavily.
<b>Destruction</b>	Some sort of grudge is the motive behind such attacks. For example, unhappy employees attack their own organization, whereas terrorists strike at much bigger levels. For example, in the year 2000, there was an attack against popular Internet sites such as Yahoo!, CNN, eBay, Buy.com, Amazon.com and e*Trade where authorized users of these sites failed to log in or access these sites.
<b>Identity theft</b>	This is best understood with a quote from Bruce Schneier: <i>Why steal from someone when you can just become that person?</i> In other words, an attacker does not steal anything from a legitimate user – he <i>becomes</i> that legitimate user! For example, it is much easier to manage to get the password of someone else's bank account or to actually be able to get a credit card on someone else's name. Then that privilege can be misused until it gets detected.
<b>Intellectual property theft</b>	Intellectual property theft ranges from stealing companies' trade secrets, databases, digital music and videos, electronic documents and books, software and so on.
<b>Brand theft</b>	It is quite easy to set up fake Web sites that look like real Web sites. How would a common user know if she is visiting the HDFC Bank site or an attacker's site? Innocent users end up providing their secrets and personal details on these fake sites to the attackers. The attackers use these details to then access the real site, causing an <i>identity theft</i> .

## 1.5.2 Attacks: A Technical View

From the technical point of view, we can classify the types of attacks on computers and network systems into two categories for better understanding: (a) Theoretical concepts behind these attacks and (b) Practical approaches used by the attackers. Let us discuss these one-by-one.

**Theoretical Concepts** As we discussed earlier, the principles of security face threat from various attacks. These attacks are generally classified into four categories, as mentioned earlier. They are:

- **Interception** – Discussed in the context of *confidentiality*, earlier. It means that an unauthorized party has gained access to a resource. The party can be a person, program or computer-based system. Examples of interception are copying of data or programs and listening to network traffic.

- Fabrication – Discussed in the context of *authentication*, earlier. This involves creation of illegal objects on a computer system. For example, the attacker may add fake records to a database.
- Modification – Discussed in the context of *integrity*, earlier. For example the attacker may modify the values in a database.
- Interruption – Discussed in the context of *availability*, earlier. Here, the resource becomes unavailable, lost or unusable. Examples of interruption are causing problems to a hardware device, erasing program, data or operating system components.

These attacks are further grouped into two types: **passive attacks** and **active attacks**, as shown in Fig. 1.10.

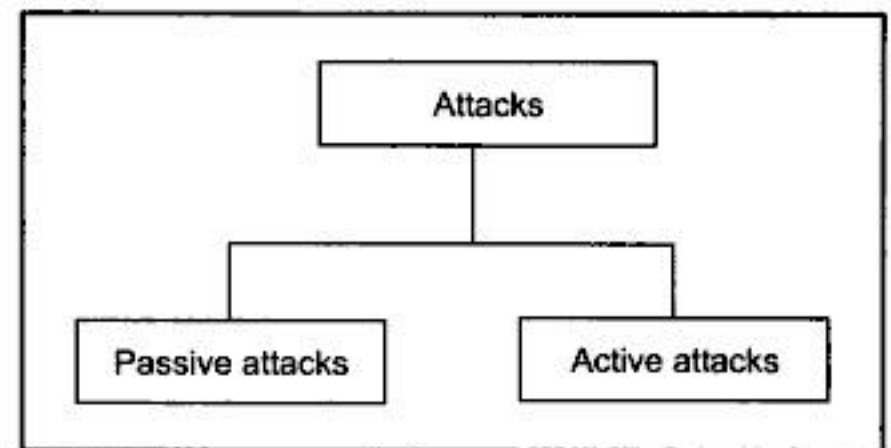
Let us discuss these two types of attacks now.

**Passive attacks** *Passive attacks* are those, wherein the attacker indulges in eavesdropping or monitoring of data transmission. In other words, the attacker aims to obtain information that is in transit. The term *passive* indicates that the attacker does not attempt to perform any modifications to

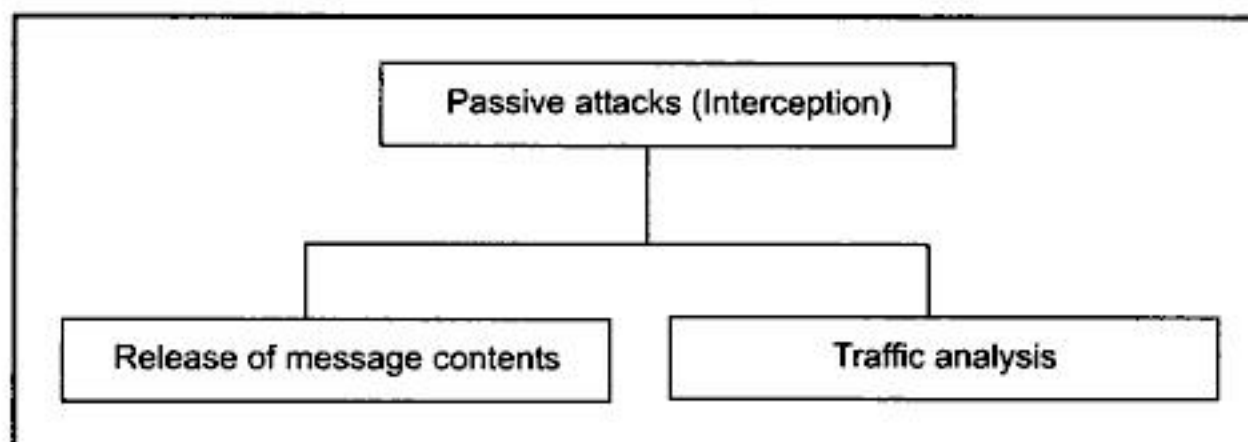
the data. In fact, this is also why passive attacks are harder to detect. Thus, the general approach to deal with passive attacks is to think about prevention, rather than detection or corrective actions.

*Passive attacks do not involve any modifications to the contents of an original message.*

Figure 1.11 shows further classification of passive attacks into two sub-categories. These categories are namely, **release of message contents** and **traffic analysis**.



┆ Fig. 1.10 *Types of attacks*



┆ Fig. 1.11 *Passive attacks*

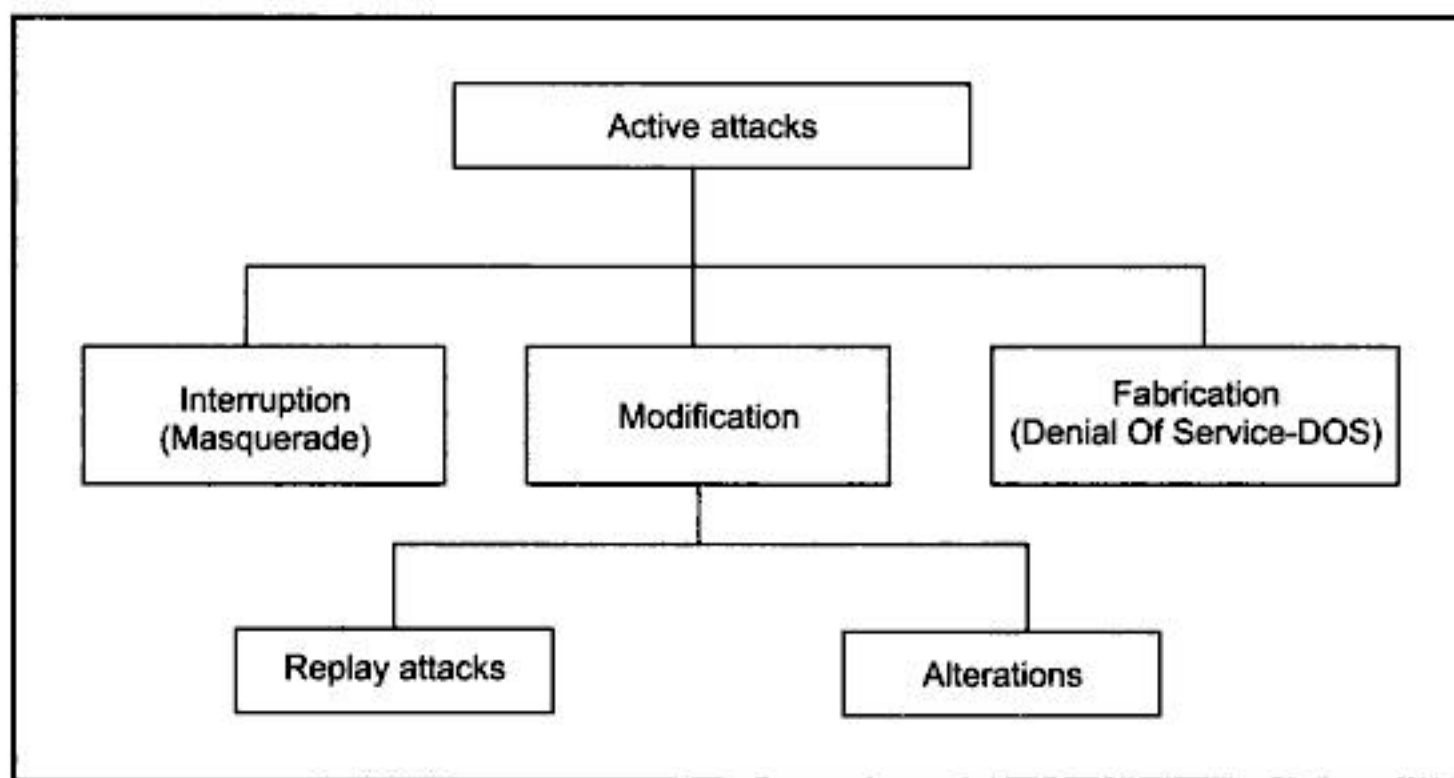
*Release of message contents* is quite simple to understand. When we send a confidential email message to our friend, we desire that only she be able to access it. Otherwise, the contents of the message are released against our wishes to someone else. Using certain security mechanisms, we can prevent *release of message contents*. For example, we can encode messages using a code language, so that only the desired parties understand the contents of a message, because only they know the code language. However, if many such messages are passing through, a passive attacker could try to figure out similarities between them to come up with some sort of pattern that provides her some clues regarding the communication that is taking place. Such attempts of analyzing (encoded) messages to come up with likely patterns are the work of the *traffic analysis* attack.

**Active attacks** Unlike *passive attacks*, the *active attacks* are based on modification of the original message in some manner or the creation of a false message. These attacks cannot be prevented easily. However, they can be detected with some effort and attempts can be made to recover from them. These attacks can be in the form of interruption, modification and fabrication.

*In active attacks, the contents of the original message are modified in some way.*

- Trying to pose as another entity involves **masquerade** attacks.
- Modification attacks can be classified further into **replay attacks** and **alteration of messages**.
- Fabrication causes **Denial Of Service (DOS)** attacks.

This classification is shown in Fig. 1.12.



┆ Fig. 1.12 Active attacks

*Masquerade* is caused when an unauthorized entity pretends to be another entity. As we have seen, user C might pose as user A and send a message to user B. User B might be led to believe that the message indeed came from user A. In masquerade attacks, an entity poses as another entity. In masquerade attacks, usually some other forms of active attacks are also embedded. As an instance, the attack may involve capturing the user's authentication sequence (e.g. user ID and password). Later, those details can be replayed to gain illegal access to the computer system.

In a *replay attack*, a user captures a sequence of events or some data units and re-sends them. For instance, suppose user A wants to transfer some amount to user C's bank account. Both users A and C have accounts with bank B. User A might send an electronic message to bank B, requesting for the funds transfer. User C could capture this message and send a second copy of the same to bank B. Bank B would have no idea that this is an unauthorized message and would treat this as a second and *different*, funds transfer request from user A. Therefore, user C would get the benefit of the funds transfer twice: once authorized, once through a replay attack.

*Alteration of messages* involves some change to the original message. For instance, suppose user A sends an electronic message *Transfer \$1000 to D's account* to bank B. User C might capture this and change it to *Transfer \$10000 to C's account*.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

# Cryptography: Concepts and Techniques

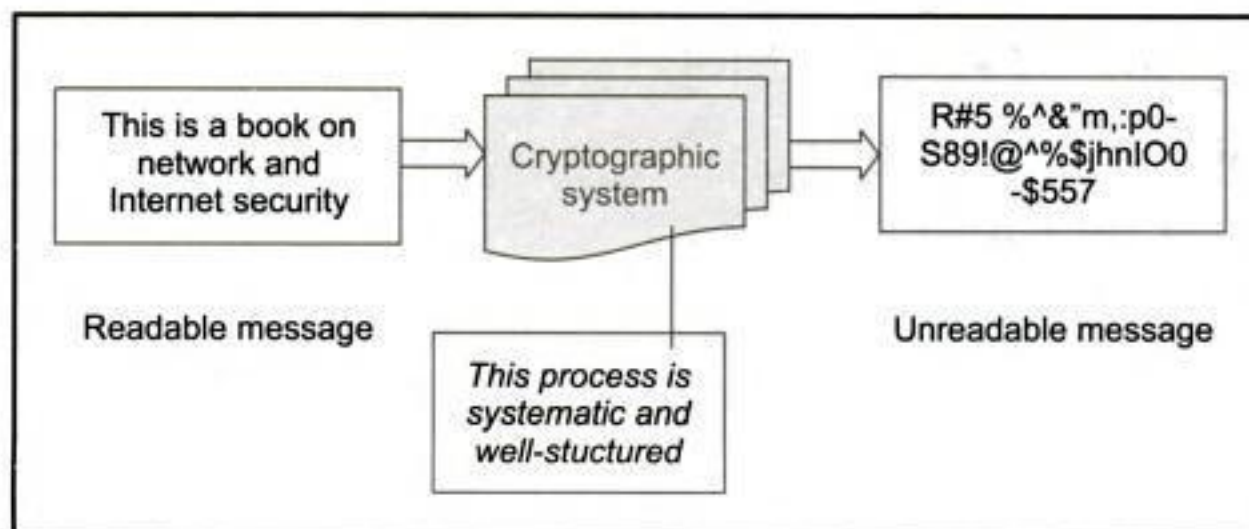


## 2.1 Introduction

This chapter introduces the basic concepts in **cryptography**. Our aim will be to demystify all the complicated terms related to this technology. After we are through with this chapter, we shall be ready to understand computer-based security solutions and issues that follow in later chapters.

*Cryptography is the art and science of achieving security by encoding messages to make them non-readable.*

Figure 2.1 shows the conceptual view of cryptography.

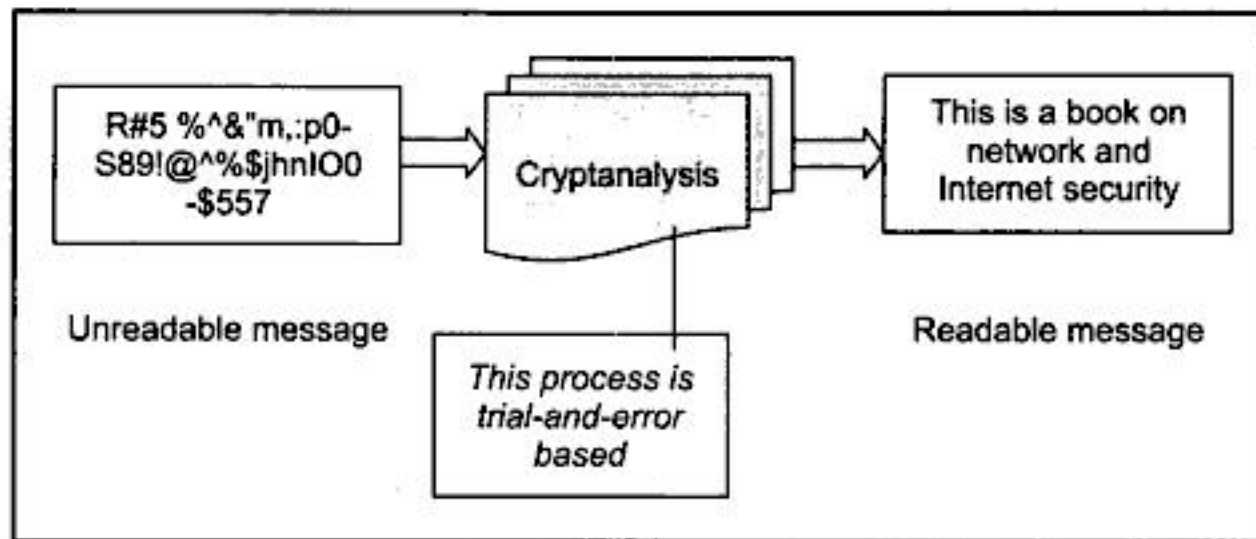


┆ Fig. 2.1 Cryptographic system

Some more terms need to be introduced in this context.

*Cryptanalysis is the technique of decoding messages from a non-readable format back to readable format without knowing how they were initially converted from readable format to non-readable format.*

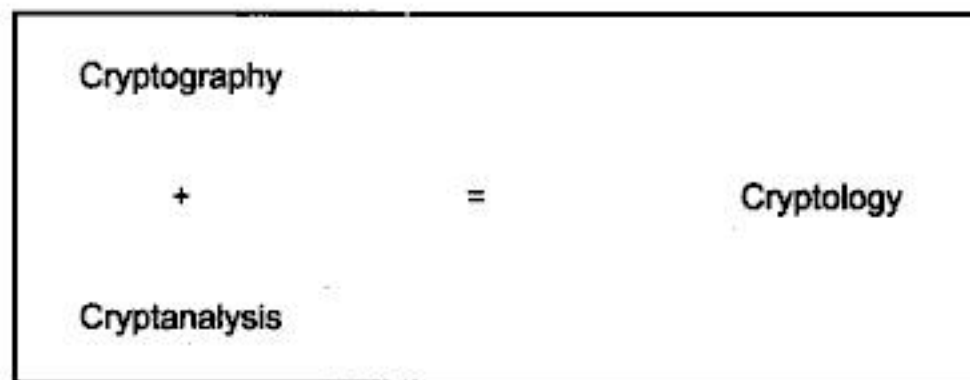
In other words, it is like *breaking a code*. This concept is shown in Fig. 2.2.



┆ Fig. 2.2 Cryptanalysis

*Cryptology* is a combination of cryptography and cryptanalysis.

This concept is shown in Fig. 2.3.



┆ Fig. 2.3 Cryptography + Cryptanalysis = Cryptology

In the early days, cryptography used to be performed by using manual techniques. The basic framework of performing cryptography has remained more or less the same, of course, with a lot of improvements in the actual implementation. More importantly, computers now perform these cryptographic functions/algorithms, thus making the process a lot faster and secure. This chapter, however, discusses the basic methods of achieving cryptography without referring to computers.

The basic concepts in cryptography are introduced first. We then proceed to discuss how we can make messages illegible and thus, secure. This can be done in many ways. We discuss all these approaches in this chapter. Modern computer-based cryptography solutions have actually evolved based on these premises. This chapter touches upon all these cryptographic algorithms. We also discuss the relative advantages and disadvantages of the various algorithms, as and when applicable.

Some cryptographic algorithms are very trivial to understand, replicate and therefore, crack. Some other cryptographic algorithms are highly complicated and therefore, difficult to crack. The rest are somewhere in the middle. A detailed discussion of these is highly essential in cementing our concepts that we shall keep referring to when we actually discuss computer-based cryptography solutions in later chapters.



## 2.2 Plain Text and Cipher Text

Any communication in the language that we speak – that is the human language, takes the form of **plain text** or **clear text**. That is, a message in plain text can be understood by anybody knowing the language as long as the message is not codified in any manner. For instance, when we speak with our family members, friends or colleagues, we use plain text because we do not want to hide anything from them. Suppose I say “Hi Anita”, it is plain text because both Anita and I know its meaning and intention. More significantly, anybody in the same room would also get to hear these words and would know that I am greeting Anita.

Notably, we also use plain text during electronic conversations. For instance, when we send an email to someone, we compose the email message using English (or these days, another) language. For instance, I can compose the email message as shown in Fig. 2.4.

Hi Amit

Hope you are doing fine. How about meeting at the train station this Friday at 5 pm?  
Please let me know if it is ok with you.

Regards.

Atul

┆ Fig. 2.4 Example of a plain text message

Now, not only Amit, but also any other person who reads this email would know what I have written. As before, this is simply because I am not using any codified language here. I have composed my email message using plain English. This is another example of plain text, albeit in written form.

*Clear text or plain text signifies a message that can be understood by the sender, the recipient and also by anyone else who gets an access to that message.*

In normal life, we do not bother much about the fact that someone could be overhearing us. In most cases, that makes little difference to us because the person overhearing us can do little damage by using the overheard information. After all, we do not reveal many secrets in our day-to-day lives.

However, there are situations where we are concerned about the secrecy of our conversations. For instance, suppose that I am interested in knowing my bank account's balance and hence I call up my phone banker from my office. The phone banker would generally ask a secret question (e.g. What is your grandmother's maiden name?) whose answer only I know. This is to ascertain that someone else is not posing as me. Now, when I give the answer to the secret question (e.g. Leela), I generally speak in low voice or better yet, initially call up from a phone that is isolated. This ensures that only the intended recipient (the phone banker) gets to know the correct answer.

On the same lines, suppose that my email to my friend Amit shown earlier is confidential for some reason. Therefore, I do not want anyone else to understand what I have written even if she is able to access the email by using some means, before it reaches Amit. How do I ensure this? This is exactly the problem that small children face. Many times, they want to communicate in such a manner that their little secrets are hidden from the elderly. What do they do in order to achieve this? Usually the simplest

trick that they use is a code language. For instance, they replace each alphabet in their conversation with another one. As an example, they replace each alphabet with the alphabet that is actually three alphabets down the order. So, each A will be replaced by D, B will be replaced by E, C will be replaced by F and so on. To complete the cycle, each W will be replaced by Z, each X will be replaced by A, each Y will be replaced by B and each Z will be replaced by C. We can summarize this scheme as shown in Fig. 2.5. The first row shows the original alphabets and the second row shows what each original alphabet will be replaced with.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

┆ Fig. 2.5 A scheme for codifying messages by replacing each alphabet with an alphabet three places down the line

Thus, using the scheme of replacing each alphabet with the one that is three places down the line, a message *I love you shall* become *L ORYH BRX* as shown in Fig. 2.6.

I		L	O	V	E		Y	O	U
L		O	R	Y	H		B	R	X

┆ Fig. 2.6 Codification using the alphabet replacement scheme

Of course, there can be many variants of such a scheme. It is not necessary to replace each alphabet with the one that is three places down the line. It can be the one that is four, five or more places down the line. The point is, however, that each alphabet in the original message can be replaced by another to hide the original contents of the message. The codified message is called as **cipher text**. Cipher means a code or a secret message.

*When a plain text message is codified using any suitable scheme, the resulting message is called as cipher text.*

Based on these concepts, let us put these terms into a diagrammatic representation, as shown in Fig. 2.7.

Let us now write our original email message and the resulting cipher text by using the alphabet-replacing scheme, as shown in Fig. 2.8. This will clarify the idea further.

As shown in Fig. 2.9, there are two primary ways in which a plain text message can be codified to obtain the corresponding cipher text: **Substitution** and **Transposition**.

Let us discuss these two approaches now. Note that when the two approaches are used together, we call the technique as **product cipher**.



## 2.3 Substitution Techniques

### 2.3.1 Caesar Cipher

The scheme explained earlier (of replacing an alphabet with the one three places down the line) was first proposed by Julius Caesar and is termed as **Caesar Cipher**. It was the first example of substitution cipher. In the substitution cipher technique, the characters of a plain text message are replaced by other



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

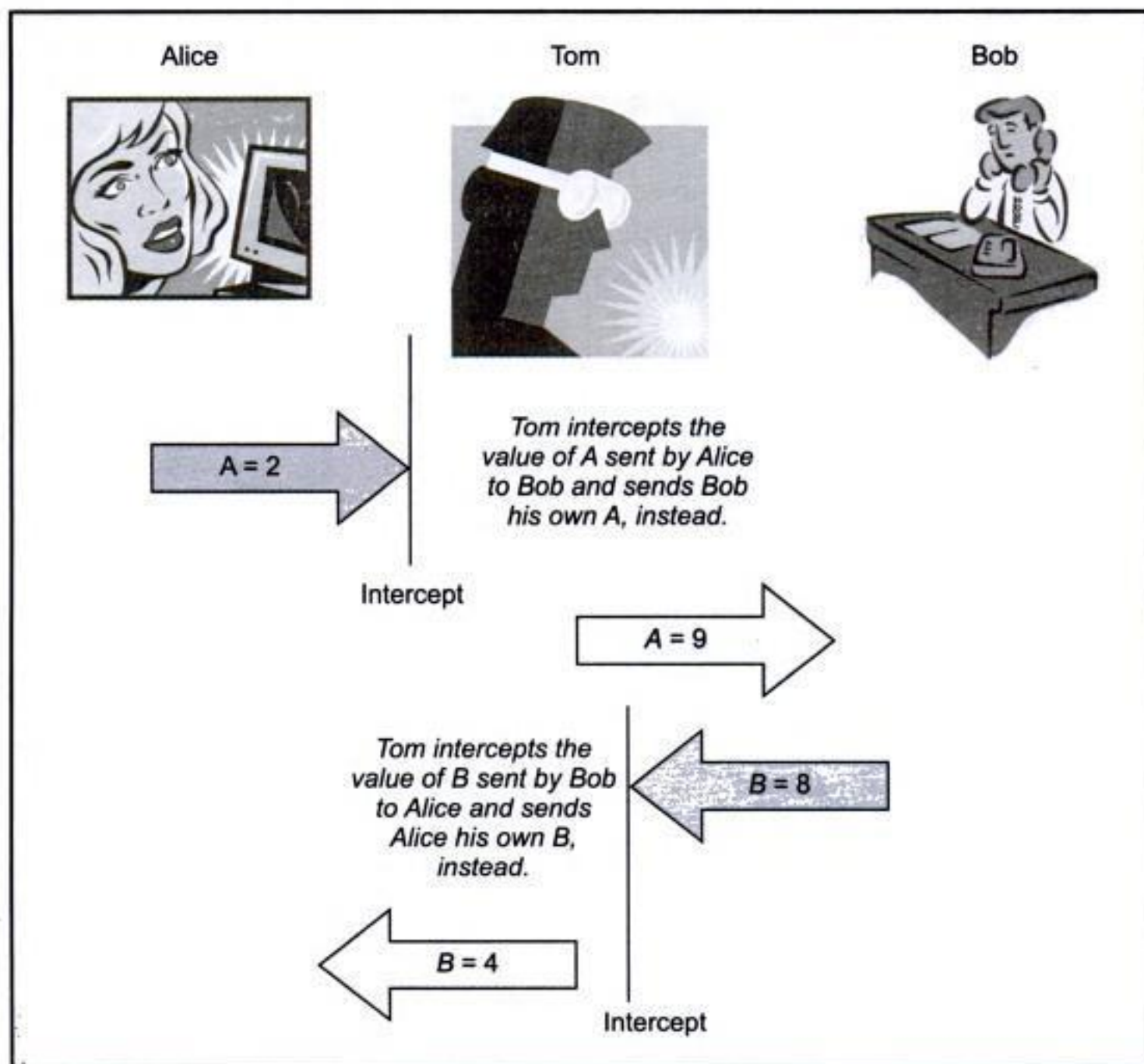


You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

<p>Alice</p> $A = g^x \text{ mod } n$ $= 7^3 \text{ mod } 11$ $= 343 \text{ mod } 11$ $= 2$	<p>Tom</p> $A = g^x \text{ mod } n$ $= 7^8 \text{ mod } 11$ $= 5764801 \text{ mod } 11$ $= 9$ $B = g^y \text{ mod } n$ $= 7^6 \text{ mod } 11$ $= 117649 \text{ mod } 11$ $= 4$	<p>Bob</p> $B = g^y \text{ mod } n$ $= 7^9 \text{ mod } 11$ $= 40353607 \text{ mod } 11$ $= 8$
---	--	--

┆ Fig. 2.55 *Man-in-the-middle attack - Part III*

5. Now, the real drama begins, as shown in Fig. 2.56.



┆ Fig. 2.56 *Man-in-the-middle attack - Part IV*

As shown in the figure, the following things happen:

- Alice sends her A (i.e. 2) to Bob. Tom intercepts it and instead, sends his A (i.e. 9) to Bob. Bob has no idea that Tom had hijacked Alice's A and has instead given his A to Bob.

- (b) In return, Bob sends his B (i.e. 8) to Alice. As before, Tom intercepts it and instead, sends his B (i.e. 4) to Alice. Alice thinks that this B came from Bob. She has no idea that Tom had intercepted the transmission from Bob and changed B.
- (c) Therefore, at this juncture, Alice, Tom and Bob have the values of A and B as shown in Fig. 2.57.

Alice	Tom	Bob
$A = 2, B = 4^*$	$A = 2, B = 8$	$A = 9^*, B = 8$
(Note: * indicates that these are the values after Tom hijacked and changed them.)		

└ Fig. 2.57 Man-in-the-middle attack - Part V

6. Based on these values, all the three persons now calculate their keys as shown in Fig. 2.58. We will notice that Alice calculates only K1, Bob calculates only K2, whereas Tom calculates both K1 and K2. Why does Tom need to do this? We shall discuss that soon.

Alice	Tom	Bob
$K1 = B^x \text{ mod } n$	$K1 = B^x \text{ mod } n$	$K2 = A^y \text{ mod } n$
$= 4^3 \text{ mod } 11$	$= 8^8 \text{ mod } 11$	$= 9^9 \text{ mod } 11$
$= 64 \text{ mod } 11$	$= 16777216 \text{ mod } 11$	$= 387420489 \text{ mod } 11$
$= 9$	$= 5$	$= 5$
	$K2 = A^y \text{ mod } n$	
	$= 2^6 \text{ mod } 11$	
	$= 64 \text{ mod } 11$	
	$= 9$	

└ Fig. 2.58 Man-in-the-middle attack - Part VI

Let us now revisit the question as to why Tom needs two keys. This is because at one side, Tom wants to communicate with Alice securely using a shared symmetric key (9) and on the other hand, he wants to communicate with Bob securely using a *different* shared symmetric key (5). Only then can he receive messages from Alice, view/manipulate them and forward them to Bob and vice versa. Unfortunately for Alice and Bob, both will (incorrectly) believe that they are directly communicating with each other. That is, Alice will feel that the key 9 is shared between her and Bob, whereas Bob will feel that the key 5 is shared between him and Alice. Actually, what is happening is, Tom is sharing the key 5 with Alice and 5 with Bob!

This is also the reason why Tom needed both sets of the secret variables  $x$  and  $y$ , as well as later on, the non-secret variables  $A$  and  $B$ .

As we can see, the *man-in-the-middle attack* can work against the Diffie–Hellman key exchange algorithm, causing it to fail. This is plainly because the *man-in-the-middle* makes the actual communicators believe that they are talking to each other, whereas they are actually talking to the *man-in-the-middle*, who is talking to each of them!

This attack can be prevented if Alice and Bob authenticate each other before beginning to exchange information. This proves to Alice is Bob is indeed Bob and not someone else (e.g. Tom) posing as Bob. Similarly, Bob can also get convinced that Alice is genuine as well.

However, not everything is lost. If we think deeply and try to come up with the scheme that will still work, an alternative emerges: namely and **asymmetric key operation**.

### 2.6.3 Asymmetric Key Operation

In this scheme, A and B do not have to jointly approach T for a lock-and-key pair. Instead, B alone approaches T, obtains a lock and a key (K1) that can seal the lock and sends the lock and key K1 to A. B tells A that A can use that lock and key to seal the box before sending the sealed box to B. How can B open the lock, then?

An interesting property of this scheme is that B possesses a *different but related* key (K2), which is obtained by B from T along with the lock and key K1, only which can open the lock. It is guaranteed that no other key and of course, including the one used by A (i.e. K1) for locking, can open the lock. Since one key (K1) is used for locking and *another, different* key (K2) is used for unlocking; we will call this scheme as *asymmetric key operation*. Also, T is clearly defined here as a **trusted third party**. T is certified as a highly trustworthy and efficient agency by the government.

This means that B possesses a **key pair** (i.e. two keys K1 and K2). One key (i.e. K1) can be used for locking and only the corresponding other key (i.e. K2) from the key pair can be used for unlocking. Thus B can send the lock and key K1 to anybody (e.g. A) who wants to send anything securely to B. B would request the sender (e.g. A) to use that lock and key K1 to seal the contents. B can then open the seal using the key K2. Since the key K1 is meant for locking and is available to the general public, we shall call K1 as **public key**. Note that K1 need not be secret – in fact, it *should not be* secret! Thus, unlike what happens in the case of symmetric key operation, the (locking) key need not be guarded secretly now. The other key K2 is meant for unlocking and is strictly held secret/private by A. Therefore, we shall call it as **private key** or **secret key**.

This is shown in Fig. 2.59.

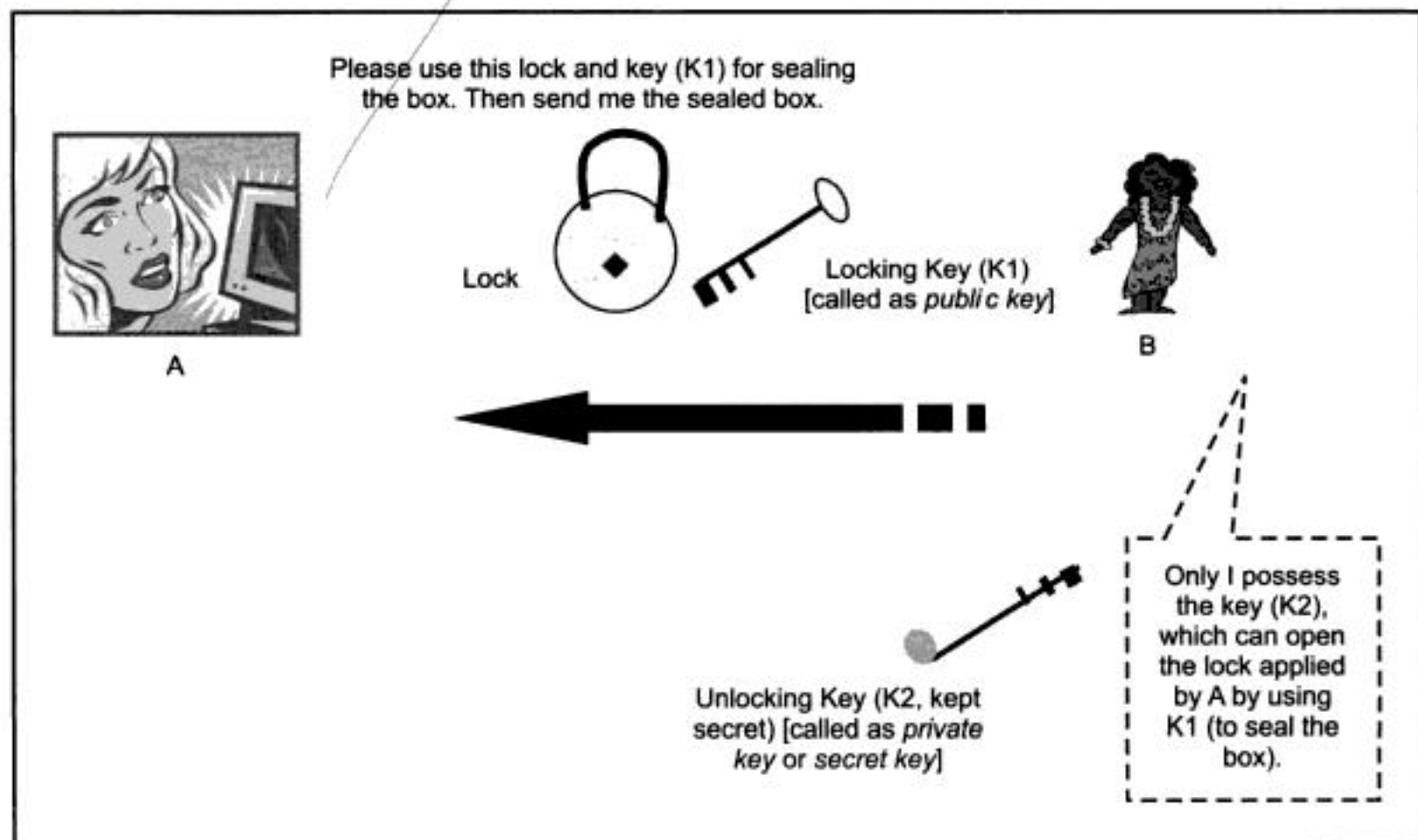


Fig. 2.59 Use of key pair



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





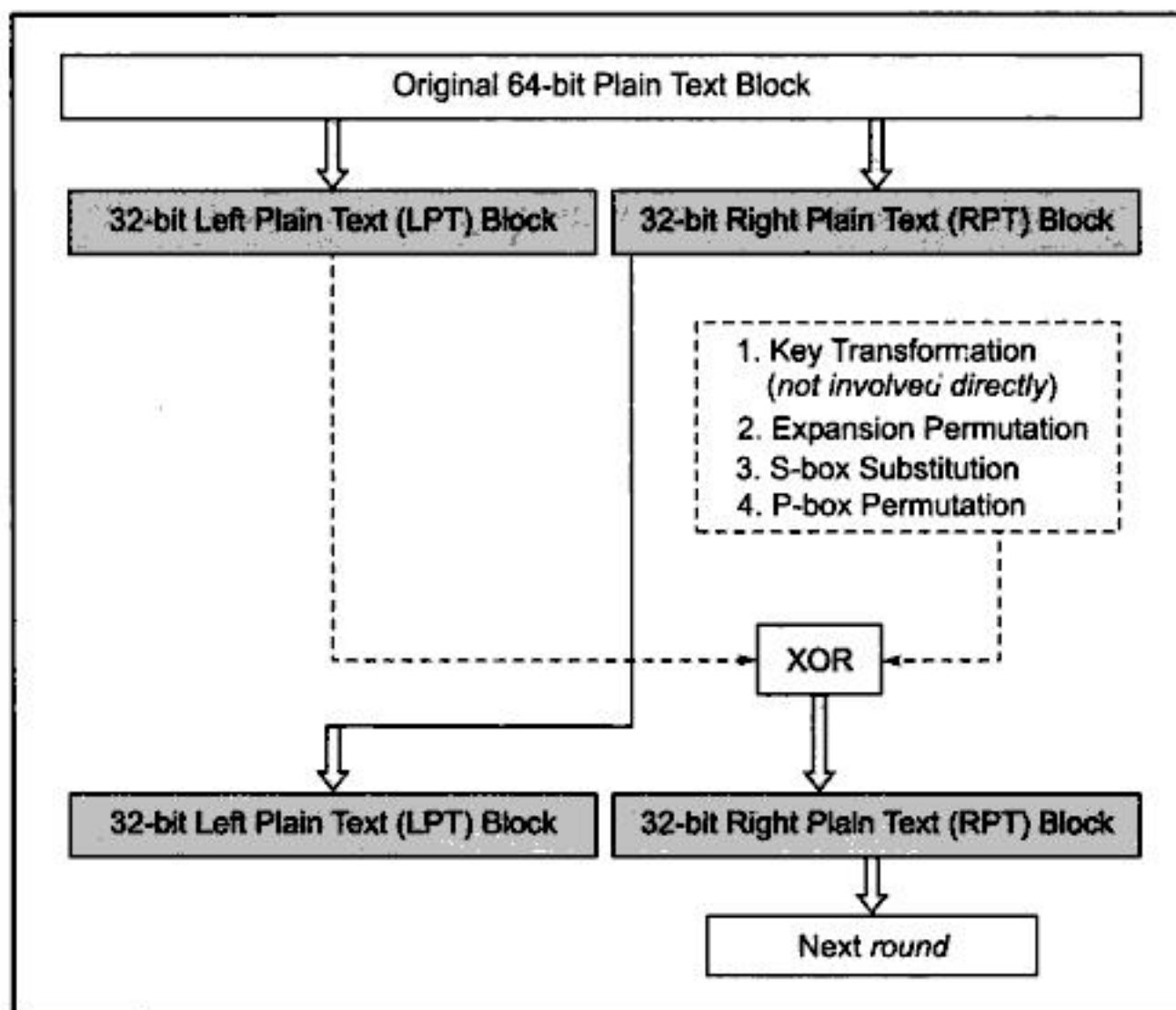
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

called as **P-box Permutation**. The P-box is shown in Fig. 3.35. For example, a 16 in the first block indicates that the bit at position 16 of the original input moves to bit at position 1 in the output and a 10 in the block number 16 indicates that the bit at the position 10 of the original input moves to bit at the position 16 in the output.

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

┆ Fig. 3.35 P-box permutation

**Step 5: XOR and Swap** Note that we have been performing all these operations only on the 32-bit right half portion of the 64-bit original plain text (i.e. on the RPT). The left half portion (i.e. LPT) was untouched so far. At this juncture, the left half portion of the initial 64-bit plain text block (i.e. LPT) is XORed with the output produced by P-box permutation. The result of this XOR operation becomes the new right half (i.e. RPT). The old right half (i.e. RPT) becomes the new left half, in a process of swapping. This is shown in Fig. 3.36.



┆ Fig. 3.36 XOR and swap

**Final Permutation** At the end of the 16 rounds, the **Final Permutation** is performed (only once). This is a simple transposition, based on Fig. 3.37. For instance, the 40<sup>th</sup> input bit takes the position of the 1<sup>st</sup> output bit and so on.

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

┆ Fig. 3.37 Final permutation

The output of the Final Permutation is the 64-bit encrypted block.

**DES decryption** From the above discussion of DES, we might get a feeling that it is an extremely complicated encryption scheme, therefore, the decryption using DES would employ a completely different approach. To most people's surprise, the same algorithm used for encryption in DES also works for decryption! The values of the various tables and the operations as well as their sequence are so carefully chosen that the algorithm is reversible. The only difference between the encryption and the decryption process is the reversal of key portions. If the original key  $K$  was divided into  $K1, K2, K3, \dots, K16$  for the 16 encryption rounds, then for decryption, the key should be used as  $K16, K15, K14, \dots, K1$ .

## Analyzing DES

**Use of S-boxes** The tables used for substitution, i.e. the S-boxes, in DES are kept secret by IBM. IBM maintains that it took them over 17 person years to come up with the internal design of the S-boxes. Over the years, suspicion has grown that there is some vulnerability in this aspect of DES, intentional (so that the government agencies could secretly open encrypted messages) or otherwise. Several studies keep appearing, which suggest that there is some scope for attacks on DES via the S-boxes. However, no concrete example has emerged till date.

**Key length** We have mentioned earlier that any cryptographic system has two important aspects: the cryptographic algorithm and the key. The inner workings of the DES algorithm (which we have discussed earlier) are completely known to the general public. Therefore, the strength of DES lies only in the other aspect - its key, which must be secret.

As we know, DES uses 56-bit keys. (Interestingly, the original proposal was to make use of 112-bit keys.) Thus, there are  $2^{56}$  possible keys (which is roughly equal to  $7.2 \times 10^{16}$  keys). Thus, it seems that a brute-force attack on DES is impractical. Even if we assume that to obtain the correct key, only half of the possible keys (i.e. the half of the **key space**) needs to be examined and tried out, a single computer performing one DES encryption per microsecond would require more than 1,000 years to break DES.

**Differential and Linear cryptanalysis** In 1990, Eli Biham and Adi Shamir introduced the concept of **differential cryptanalysis**. This method looks at pairs of cipher text whose plain texts have particular differences. The technique analyses the progress of these differences as the plain texts travel through the various rounds of DES. The idea is to choose pairs of plain text with fixed differences. The two plain texts can be chosen at random, as long as they satisfy specific difference conditions (which can be as simple as XOR). Then, using the differences in the resulting cipher texts, assign different likelihood to different keys. As more and more cipher text pairs are analysed, the correct key emerges.

Invented by Mitsuru Matsui, the **linear cryptanalysis** attack is based on linear approximations. If we XOR some plain text bits together, XOR some cipher text bits together and then XOR the result, we will get a single bit, which is the XOR of some of the key bits.

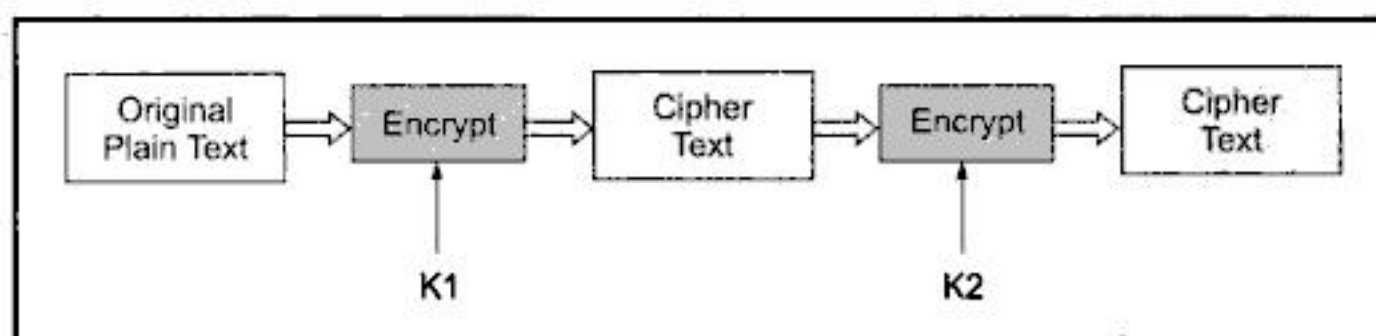
The descriptions of these attacks are quite complex and we will not discuss them here.

**Timing attacks** Timing attacks refer more to asymmetric key cryptography. However, they can also apply to symmetric key cryptography. The idea is simple: observe how long it takes for the cryptographic algorithm to decrypt different blocks of cipher text. The idea is to try and obtain either the plain text or the key used for encryption by observing these timings. In general, it would take different amounts of time to decrypt different sized cipher text blocks.

### 3.4.3 Variations of DES

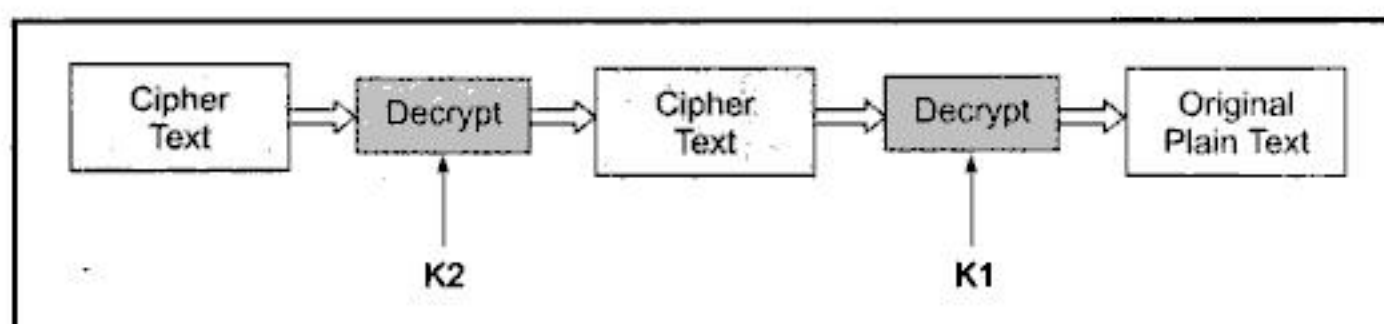
In spite of its strengths, it is generally felt that with the tremendous advances in computer hardware (processing speeds of Gigahertz and more, high memory availability at cheap prices, parallel processing capabilities, etc.), DES is susceptible to possible attacks. However, because DES is already proven to be a very competent algorithm, it would be nice to reuse DES by making it stronger by some means, rather than writing a new cryptographic algorithm. Writing a new algorithm is not easy, more so because it has to be tested sufficiently so as to be proved as a strong algorithm. Consequently, two main variations of DES have emerged, which are **Double DES** and **Triple DES**. Let us discuss them now.

**Double DES** Double DES is quite simple to understand. Essentially, it does twice what DES normally does only once. Double DES uses two keys, say  $K1$  and  $K2$ . It first performs DES on the original plain text using  $K1$  to get the encrypted text. It again performs DES on the encrypted text, but this time with the other key, i.e.  $K2$ . The final output is the encryption of encrypted text (i.e. the original plain text encrypted twice with two different keys). This is shown in Fig. 3.38.



┆ Fig. 3.38 Double DES encryption

Of course, there is no reason why double encryption cannot be applied to other cryptographic algorithms as well. However, in the case of DES, it is already quite popular, therefore, we have discussed this in the context of DES. It should also be quite simple to imagine that the decryption process would work in exactly the reverse order, as shown in Fig. 3.39.



┆ Fig. 3.39 Double DES decryption

The doubly encrypted cipher text block is first decrypted using the key  $K2$  to produce the singly encrypted cipher text. This cipher text block is then decrypted using the key  $K1$  to obtain the original plain text block.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

- ❑ Blowfish was developed by Bruce Schneier and has the reputation of being a very strong symmetric key cryptographic algorithm.
- ❑ Blowfish encrypts 64-bit blocks with a variable-length key.
- ❑ In the 1990s, the US Government wanted to standardize a cryptographic algorithm, which was to be used universally by them. It was to be called as the Advanced Encryption Standard (AES). Many proposals were submitted and after a lot of debate, an algorithm called as Rijndael was accepted.
- ❑ Rijndael supports key lengths and plain text block sizes from 128 bits to 256 bits, in the Steps of 32 bits. The key length and the length of the plain text blocks need to be selected independently. AES mandates that the plain text block size must be 128 bits and key size should be 128, 192 or 256 bits. In general, two versions of AES are used: 128-bit plain text block combined with 128-bit key block and 128-bit plain text block with 256-bit key block.



## KEY TERMS AND CONCEPTS

- ❖ Advanced Encryption Standard (AES)
- ❖ Algorithm mode
- ❖ Algorithm type
- ❖ Block cipher
- ❖ Chaining mode
- ❖ Cipher Block Chaining (CBC)
- ❖ Cipher Feedback (CFB)
- ❖ Confusion
- ❖ Counter mode
- ❖ Data Encryption Standard (DES)
- ❖ Differential cryptanalysis
- ❖ Diffusion
- ❖ Double DES
- ❖ Electronic Code Book (ECB)
- ❖ Galois field
- ❖ Group
- ❖ Initialization Vector (IV)
- ❖ International Data Encryption Algorithm (IDEA)
- ❖ Keystream
- ❖ Linear cryptanalysis
- ❖ Lucifer
- ❖ Meet-in-the-middle attack
- ❖ Output Feedback (OFB)
- ❖ RC4
- ❖ RC5
- ❖ Rijndael
- ❖ Stream cipher
- ❖ Timing attacks
- ❖ Triple DES



## PRACTICE SET



## MULTIPLE-CHOICE QUESTIONS

1. In \_\_\_\_\_, one bit of plain text is encrypted at a time.
  - (a) stream cipher
  - (b) block cipher
  - (c) both stream and block cipher
  - (d) none of the above
2. In \_\_\_\_\_, one block of plain text is encrypted at a time.
  - (a) stream cipher
  - (b) block cipher

- (c) both stream and block cipher (d) none of the above
3. \_\_\_\_\_ increases the redundancy of plain text.  
(a) Confusion (b) Diffusion  
(c) Both confusion and diffusion (d) Neither confusion nor diffusion
4. \_\_\_\_\_ works on block mode.  
(a) CFB (b) OFB  
(c) CCB (d) CBC
5. DES encrypts blocks of \_\_\_\_\_ bits.  
(a) 32 (b) 56  
(c) 64 (d) 128
6. There are \_\_\_\_\_ rounds in DES.  
(a) 8 (b) 10  
(c) 14 (d) 16
7. \_\_\_\_\_ is based on the IDEA algorithm.  
(a) S/MIME (b) PGP  
(c) SET (d) SSL
8. The actual algorithm in the AES encryption scheme is \_\_\_\_\_ .  
(a) Blowfish (b) IDEA  
(c) Rijndael (d) RC4
9. The Blowfish algorithm executes the \_\_\_\_\_ algorithm for subkey generation.  
(a) Blowfish (b) IDEA  
(c) Rijndael (d) RC4
10. In AES, the 16-byte key is expanded into \_\_\_\_\_ .  
(a) 200 bytes (b) 78 bytes  
(c) 176 bytes (d) 184 bytes
11. In IDEA, the key size is \_\_\_\_\_ .  
(a) 128 bytes (b) 128 bits  
(c) 256 bytes (d) 256 bits
12. In the \_\_\_\_\_ algorithm, once the initial key of 1-256 bytes is used to create a transformed key, the original key is discarded.  
(a) Blowfish (b) IDEA  
(c) Rijndael (d) RC4
13. There are \_\_\_\_\_ encryption rounds in RC5.  
(a) 8 (b) 12  
(c) 16 (d) 20
14. The RC5 block cipher mode is also called as \_\_\_\_\_ .  
(a) RC5 block cipher (b) RC5-CBC  
(c) RC5-CBC-Pad (d) RC5-CTS
15. The \_\_\_\_\_ step ensures that plain text is not vulnerable in block cipher mode.  
(a) encryption (b) round  
(c) initial (d) chaining



## EXERCISES

1. Distinguish between stream and block ciphers.
2. Discuss the idea of algorithm modes with detailed explanation of at least two of them.
3. Write a note on the security and possible vulnerabilities of the various algorithm modes.
4. What is an Initialization Vector (IV)? What is its significance?
5. What are the problems with symmetric key encryption?
6. What is the idea behind meet-in-the-middle attack?
7. Explain the main concepts in DES.
8. How can the same key be reused in triple DES?
9. Explain the principles of the IDEA algorithm.
10. Distinguish between differential and linear cryptanalysis.
11. Explain the subkey generation in the Blowfish algorithm.
12. Explain the usage of the S array in the case of the RC4 algorithm.
13. Discuss how encryption happen in RC5.
14. How does the one-time initialization step work in AES?
15. Explain the steps in the various rounds of AES.



## DESIGN/PROGRAMMING EXERCISES

1. Write a C program to implement the DES algorithm logic.
2. Write the same program as in Step 1, in Java.
3. Write a Java program that contains functions, which accept a key and input text to be encrypted/decrypted. This program should use the key to encrypt/decrypt the input by using the triple DES algorithm. Make use of Java cryptography package.
4. Write a C program to implement the Blowfish algorithm.
5. Write the same program in VB.NET.
6. Investigate Rijndael further and write a C program to implement the same.
7. Find out more about the vulnerabilities of DES from the Internet.
8. Write the RC4 logic in Java.
9. Try to implement the logic of the various algorithm modes in a programming language of your choice.
10. Create a visual tool that should take as input some plain text and a key and then execute an algorithm of the user's choice (e.g. DES). It should provide a visual display of the output at each stage of the encryption process.
11. Using Java cryptography, encrypt the text "Hello world" using Blowfish. Create your own key using Java *keytool*.
12. Perform the same task by using the cryptography API in .NET.
13. Examine which software products in real life use which cryptographic algorithms. Try to find out the reasons behind the choice of these algorithms.
14. Implement DES-2 and DES-3 (with two keys) using Java cryptography and try to encrypt a large block of text. Find out if there is any significant performance difference.
15. Compare the results of the above with DES-3 (with three keys).



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

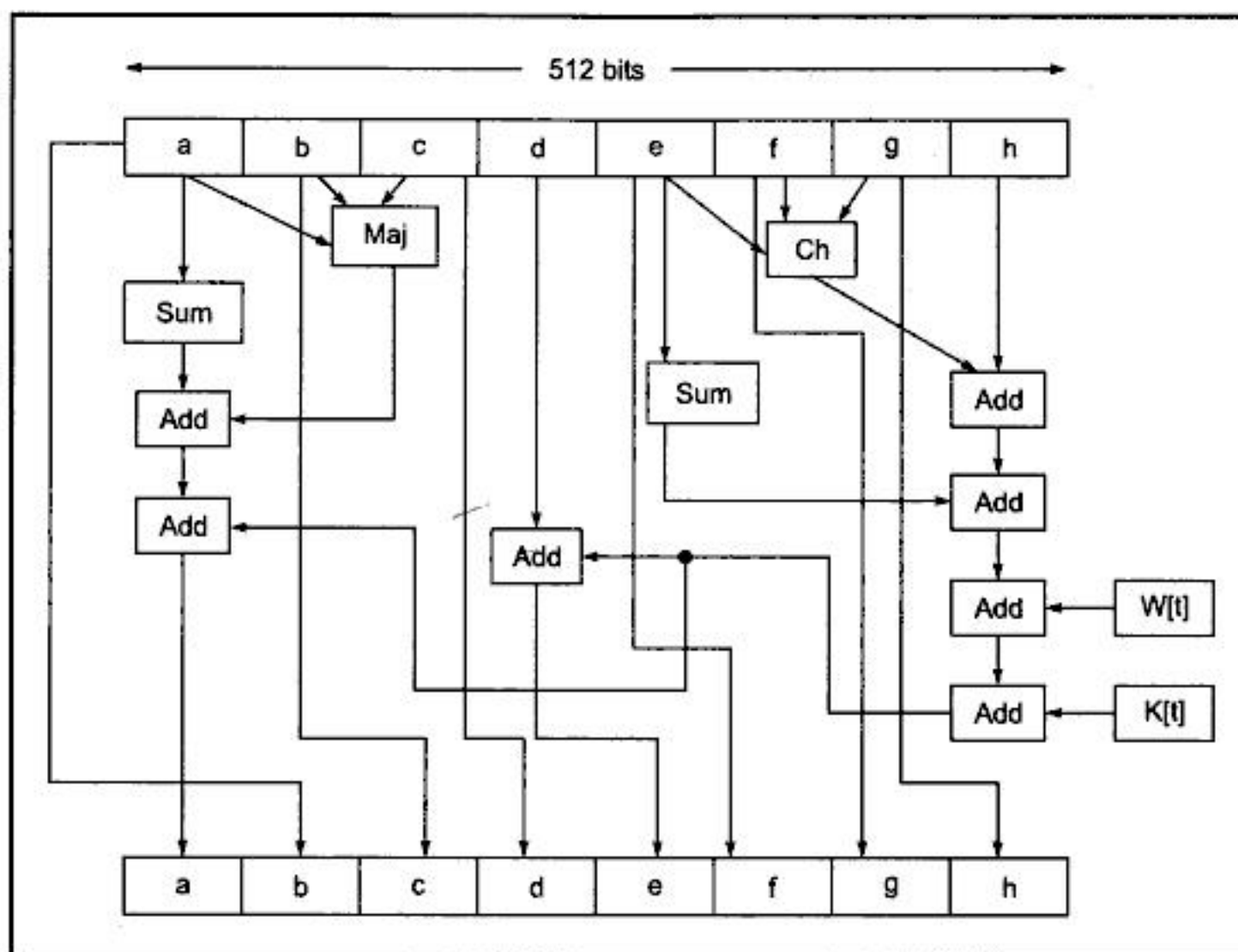


Fig. 4.33 Single SHA-512 iteration

$$\begin{aligned}
 d &= c \\
 e &= d + \text{Temp1} \\
 f &= e \\
 g &= f \\
 h &= g
 \end{aligned}$$

where:

$$\begin{aligned}
 t &= \text{Round number} \\
 \text{Ch}(e, f, g) &= (e \text{ AND } f) \text{ XOR } (\text{NOT } e \text{ AND } g) \\
 \text{Maj}(a, b, c) &= (a \text{ AND } b) \text{ XOR } (a \text{ AND } c) \text{ XOR } (b \text{ AND } c) \\
 \text{Sum}(a_i) &= \text{ROTR}(a_i \text{ by } 28 \text{ bits}) \text{ XOR } \text{ROTR}(a_i \text{ by } 34 \text{ bits}) \text{ XOR } (\text{ROTR } a_i \text{ by } 39 \text{ bits}) \\
 \text{Sum}(e_i) &= \text{ROTR}(e_i \text{ by } 14 \text{ bits}) \text{ XOR } \text{ROTR}(e_i \text{ by } 18 \text{ bits}) \text{ XOR } (\text{ROTR } e_i \text{ by } 41 \text{ bits}) \\
 \text{ROTR}(x) &= \text{Circular right shift, i.e. rotation, of the 64-bit array } x \text{ by the specified number of bits} \\
 W_t &= 64\text{-bit word derived from the current 512-bit input block} \\
 K_t &= 64\text{-bit additive constant} \\
 + \text{ (or Add)} &= \text{Addition mod } 2^{64}
 \end{aligned}$$

The 64-bit word values for  $W_t$  are derived from the 1024-bit message using certain mappings, which we shall not describe here. Instead, we will simply point out this:

1. For the first 16 rounds (0 to 15), the value of  $W_t$  is equal to the corresponding word in the message block.



- For the remaining 64 steps, the value of  $W_t$  is equal to the circular left shift by one bit of the XOR of the four preceding values of  $W_t$  with two of them subjected to shift and rotate operations.

This makes the message digest more complex and difficult to break.

#### 4.6.6 Message Authentication Code (MAC)

The concept of **Message Authentication Code (MAC)** is quite similar to that of a message digest. However, there is one difference. As we have seen, a message digest is simply a fingerprint of a message. There is no cryptographic process involved in the case of message digests. In contrast, a MAC requires that the sender and the receiver should know a shared symmetric (secret) key, which is used in the preparation of the MAC. Thus, MAC involves cryptographic processing. Let us see how this works.

Let us assume that the sender A wants to send a message M to a receiver B. How the MAC processing works is shown in Fig. 4.34.

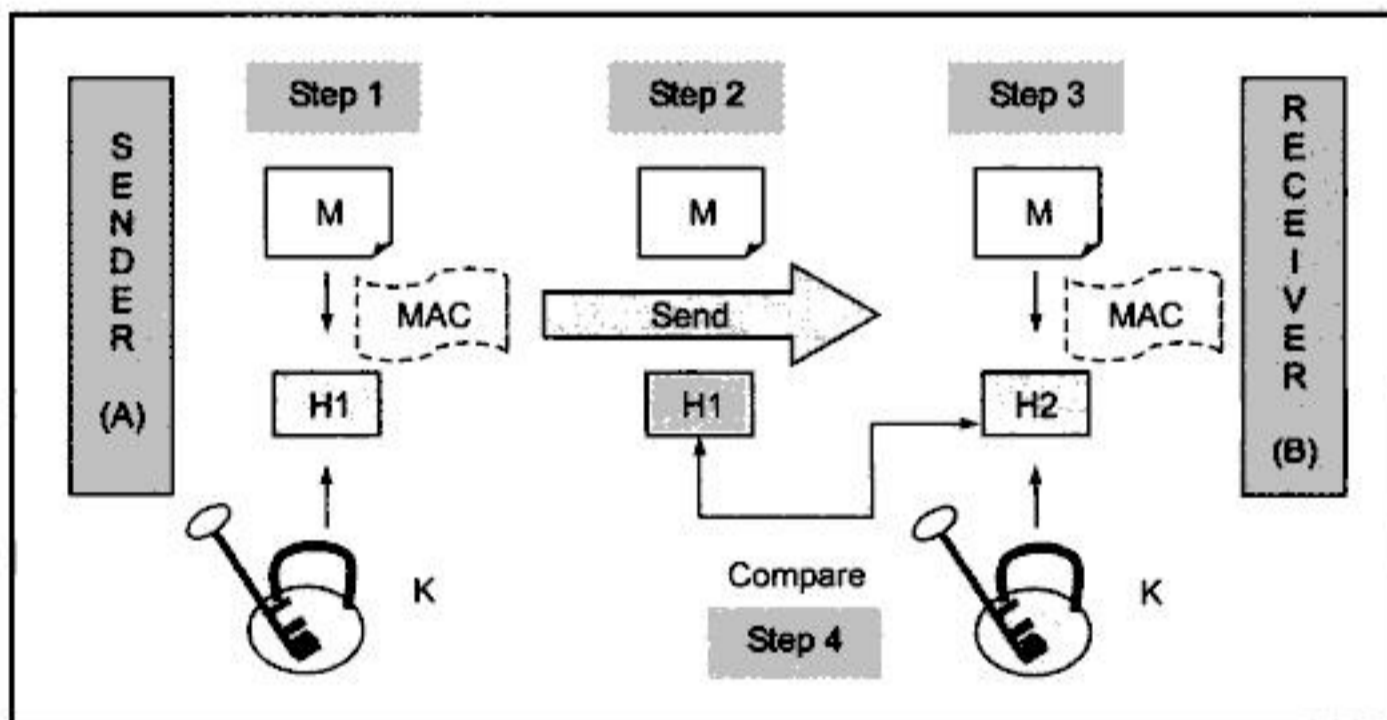


Fig. 4.34 Message authentication code (MAC)

- A and B share a symmetric (secret) key  $K$ , which is not known to anyone else. A calculates the MAC by applying key  $K$  to the message  $M$ .
- A then sends the original message  $M$  and the MAC  $H1$  to B.
- When B receives the message, B also uses  $K$  to calculate its own MAC  $H2$  over  $M$ .
- B now compares  $H1$  with  $H2$ . If the two match, B concludes that the message  $M$  has not been changed during transit. However, if  $H1 \neq H2$ , B rejects the message, realizing that the message was changed during transit.

The significances of a MAC are as follows:

- The MAC assures the receiver (in this case, B) that the message is not altered. This is because if an attacker alters the message but does not alter the MAC (in this case,  $H1$ ), then the receiver's calculation of the MAC (in this case,  $H2$ ) will differ from it. Why does the attacker then not also alter the MAC? Well, as we know, the key used in the calculation of the MAC (in this case,  $K$ ) is

assumed to be known only to the sender and the receiver (in this case, A and B). Therefore, the attacker does not know the key,  $K$  and therefore, she cannot alter the MAC.

2. The receiver (in this case, B) is assured that the message indeed came from the correct sender (in this case, A). Since only the sender and the receiver (A and B, respectively, in this case) know the secret key (in this case,  $K$ ), no one else could have calculated the MAC (in this case,  $H1$ ) sent by the sender (in this case, A).

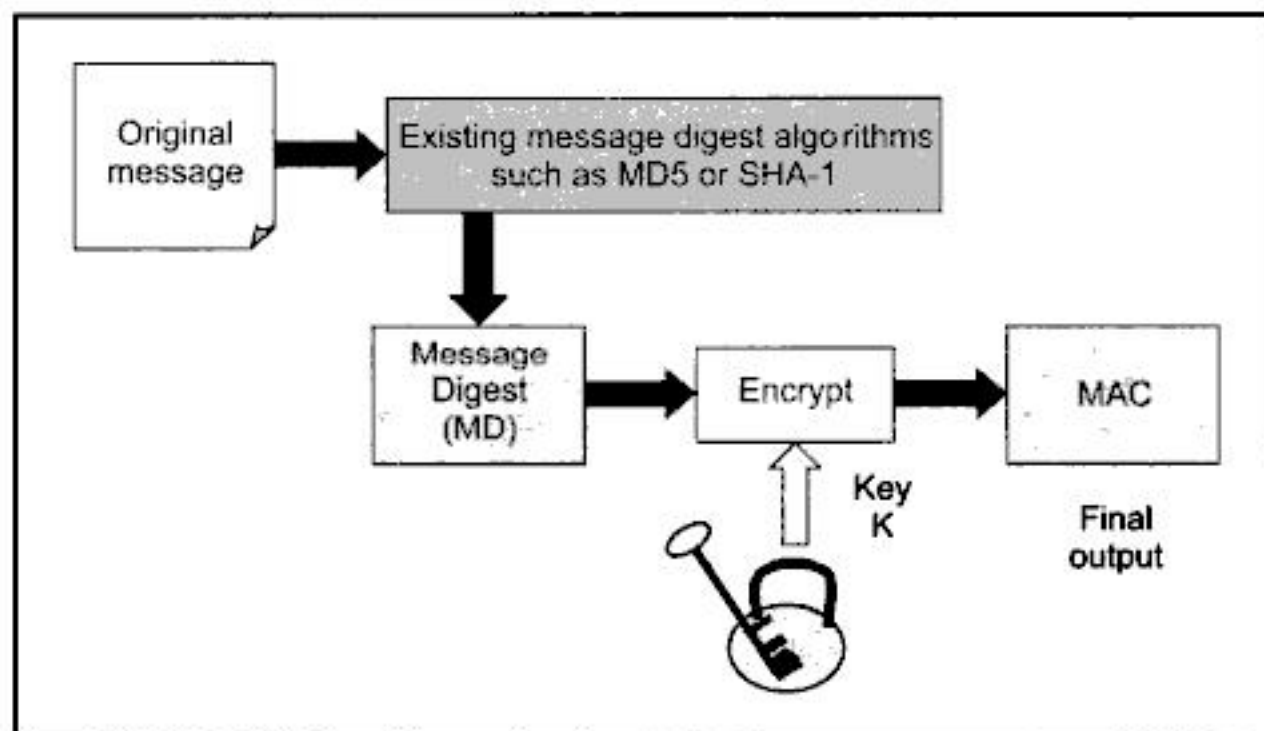
Interestingly, although the calculation of the MAC seems to be quite similar to an encryption process, it is actually different in one important respect. As we know, in symmetric key cryptography, the cryptographic process must be reversible. That is, the encryption and decryption are the mirror images of each other. However, note that in the case of MAC, both the sender and the receiver are performing encryption process only. Thus, a MAC algorithm need not be reversible – it is sufficient to be a one-way function (encryption) only.

We have already discussed two main message digest algorithms, namely MD5 and SHA-1. Can we reuse these algorithms for calculating a MAC, in their original form? Unfortunately, we cannot reuse them, because they do not involve the usage of a secret key, which is the basis of MAC. Consequently, we must have a separate practical algorithm implementation for MAC. The solution is **HMAC**, a practical algorithm to implement MAC.

#### 4.6.7 HMAC

**Introduction** HMAC stands for **Hash-based Message Authentication Code**. HMAC has been chosen as a mandatory security implementation for the Internet Protocol (IP) security and is also used in the Secure Socket Layer (SSL) protocol, widely used on the Internet.

The fundamental idea behind HMAC is to reuse the existing message digest algorithms, such as MD5 or SHA-1. Obviously, there is no point in reinventing the wheel. Therefore, what HMAC does it to work with any message digest algorithm. That is, it treats the message digest as a black box. Additionally, it uses the shared symmetric key to encrypt the message digest, which produces the output MAC. This is shown in Fig. 4.35.



┆ Fig. 4.35 HMAC concept



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

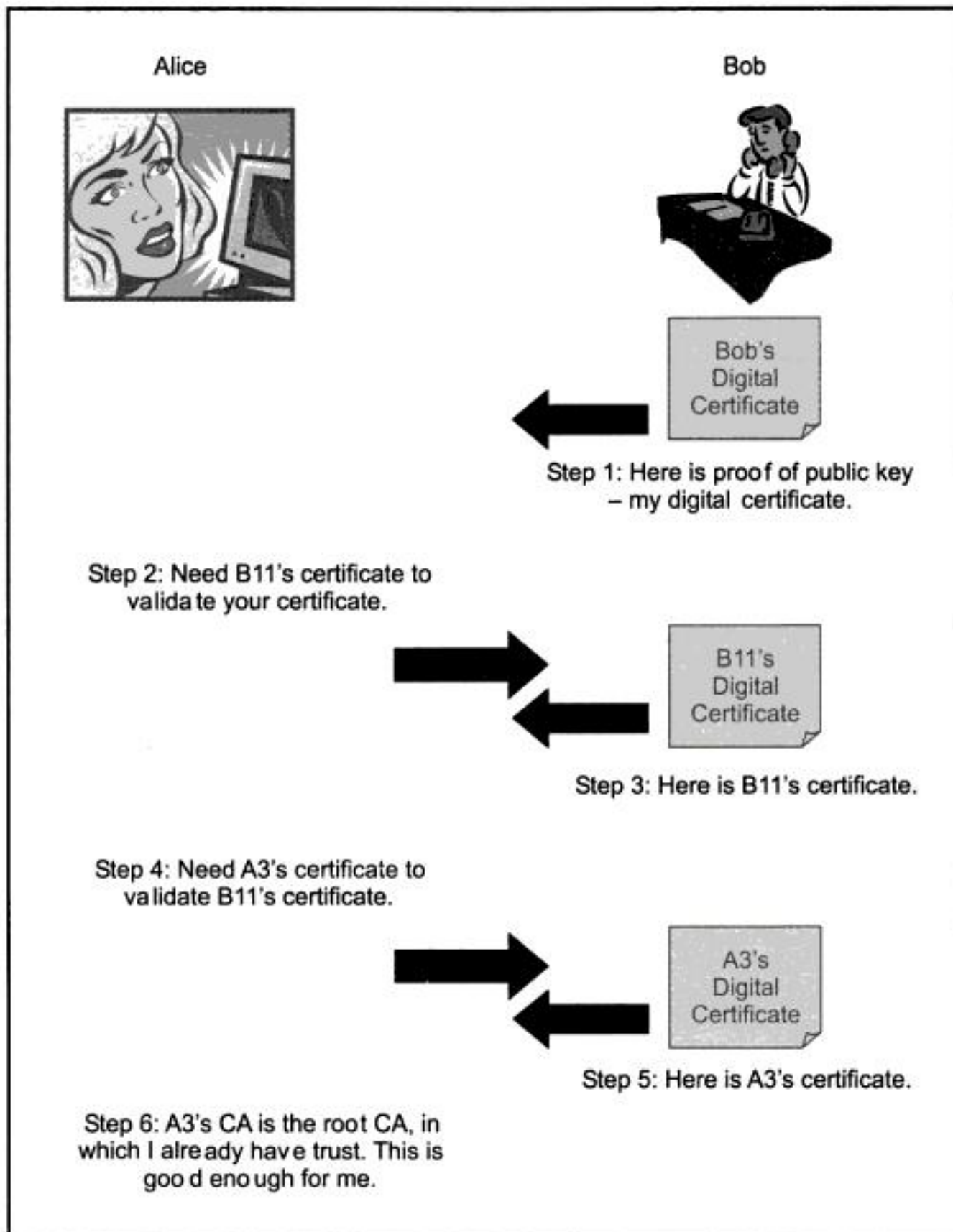


Fig. 5.24 Verification of a root CA

user in the world is quite unlikely. Instead, the concept of decentralized CAs for different countries, political organizations and businesses is far better. This allows the CAs not only to worry about a smaller population of users, but also work independently. Moreover, *cross-certification* allows CAs and end users from different PKI domains to interact.

More specifically, cross-certification certificates are issued by the CAs to create a non-hierarchical trust path. Let us understand this with an example as shown in Fig. 5.25.

As the figure shows, the root CAs of Alice and Bob are different – but they have cross-certified each other. Technically, this means that Alice's root CA has obtained a certificate for itself from Bob's root CA. Similarly, Bob's root CA has obtained a certificate for itself from Alice's root CA. Now, even if Alice's software trusts only her own root CA, it is not a problem. Since Bob's root CA is certified by

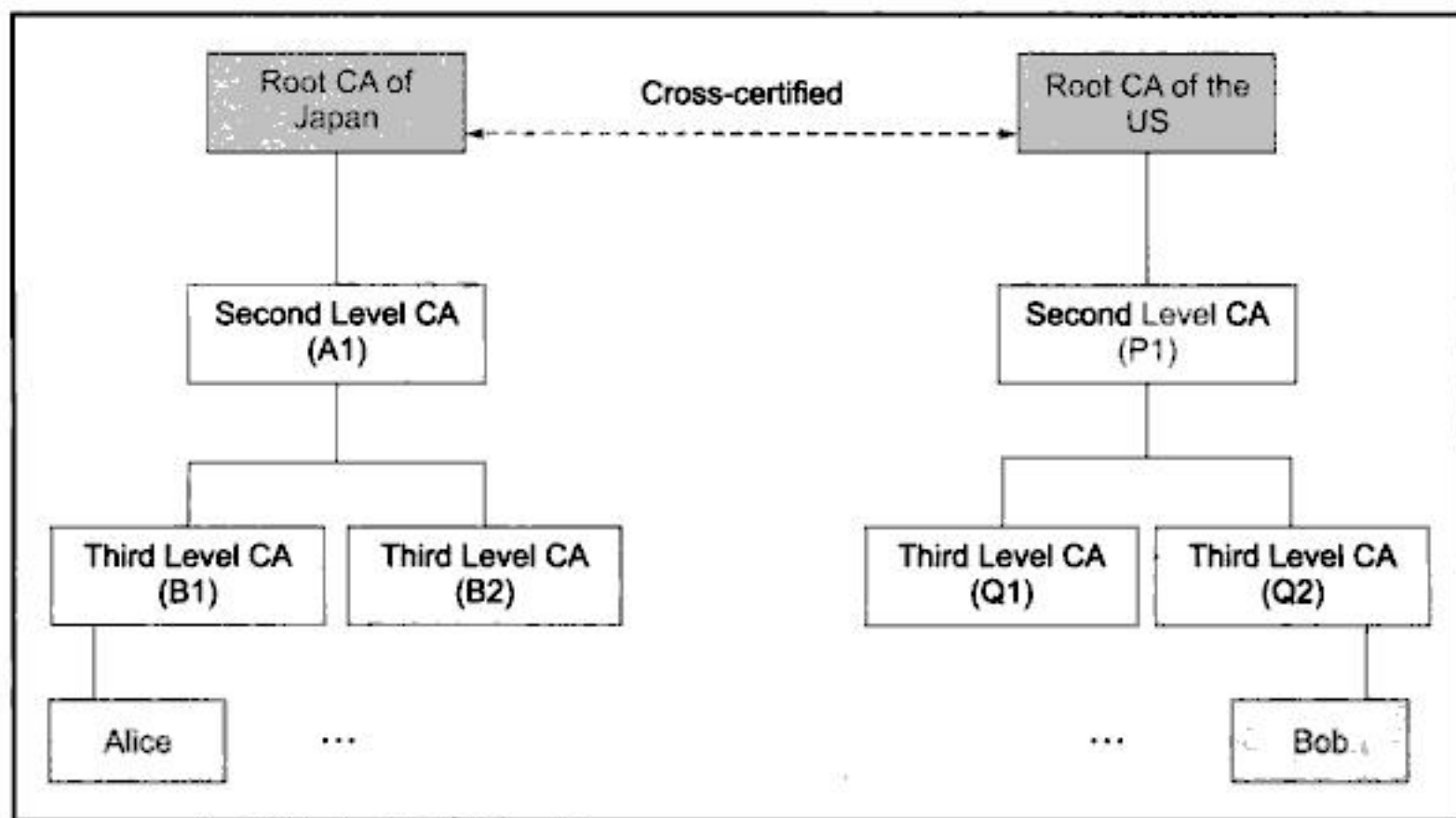


Fig. 5.25 Cross-certification of CAs

Alice's root CA, Alice can and does trust the root CA of Bob. This also means that Alice can verify Bob's certificate using the path Bob – Q2 – P1 – Bob's root CA – Alice's root CA. We shall not describe how this happens, as we have already explained this process in great detail earlier.

The morale of the story is that with the technologies of certificate hierarchies, self-signed certificates and cross-certification, virtually any user can verify any other user's digital certificate and based on that, decide to either trust it or reject it.

### 5.2.9 Certificate Revocation

**Introduction** If your credit card is lost or if it gets stolen, you would normally immediately report the loss to the concerned bank. The bank would cancel your credit card. Similarly, digital certificate can also be revoked. What can be the reasons for the revocation of a digital certificate? Some of the most common ones are as follows:

- The holder of the digital certificate reports that the private key corresponding to the public key specified in the digital certificate is compromised (i.e. someone has stolen it).
- The CA realizes that it had made some mistake while issuing a certificate.
- The certificate holder leaves a job and the certificate was issued specifically while the person was employed in that job.

In all such cases, the digital certificate of the user gets the same status as that of a stolen credit card. Therefore, the certificate must be treated as invalid. For this purpose, the certificate must be revoked. What is the process for this? Normally, just as a credit card user reports a credit card loss or theft, a certificate holder should report that a certificate should be revoked. Of course, if the user leaves an organization or indulges in an illegal act because of which the certificate needs to be revoked, the organization should initiate the process. Finally, if the CA realizes its own mistake in providing a wrong certificate, the CA initiates the certification revocation process.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

- **Content (1 or more bytes):** This field contains the parameters associated with this message, depending on the message type, as listed in Fig. 6.12.

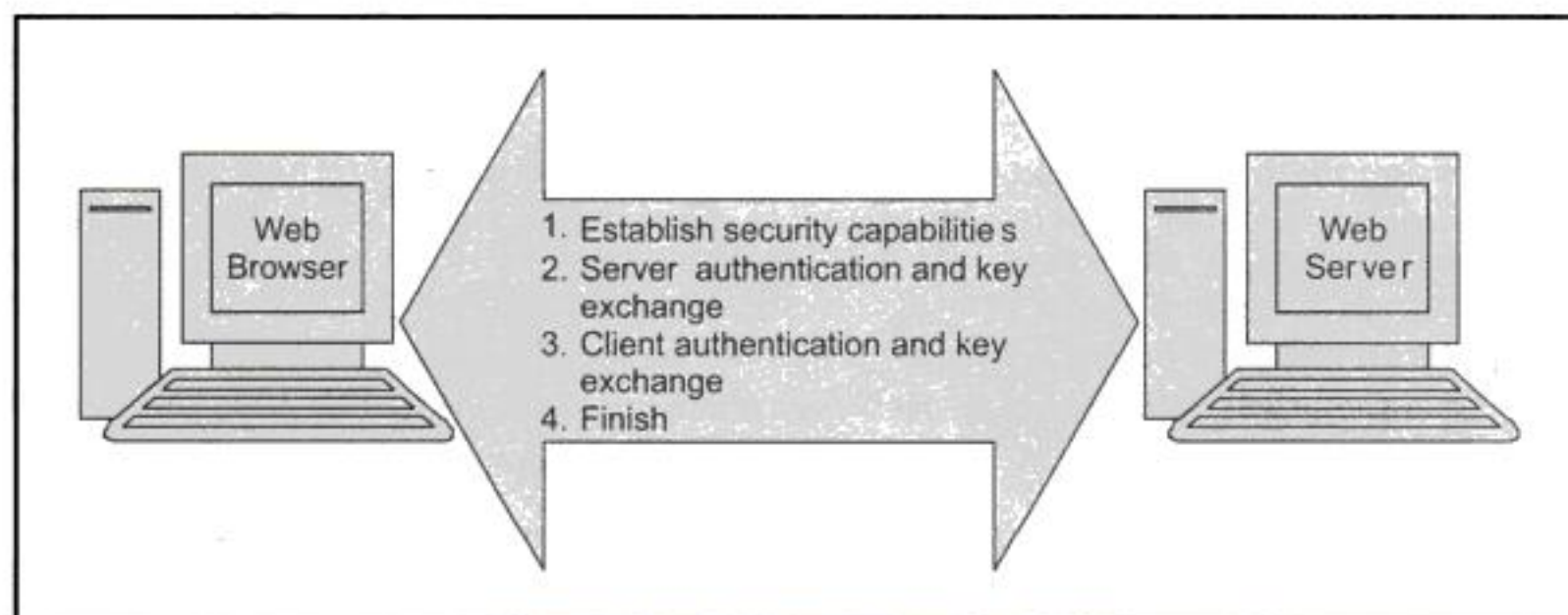
Let us now take a look at the possible messages exchanged by the client and the server in the handshake protocol, along with their corresponding parameters, as shown in Fig. 6.12.

Message Type	Parameters
Hello request	None
Client hello	Version, Random number, Session id, Cipher suite, Compression method
Server hello	Version, Random number, Session id, Cipher suite, Compression method
Certificate	Chain of X.509V3 certificates
Server key exchange	Parameters, signature
Certificate request	Type, authorities
Server hello done	None
Certificate verify	Signature
Client key exchange	Parameters, signature
Finished	Hash value

┆ Fig. 6.12 SSL handshake protocol message types

The handshake protocol is actually made up of four phases, as shown in Fig. 6.13. These phases are:

1. Establish security capabilities
2. Server authentication and key exchange
3. Client authentication and key exchange
4. Finish



┆ Fig. 6.13 SSL handshake phases

Let us now study these four phases one by one.

**Phase 1. Establish Security Capabilities** This first phase of the SSL handshake is used to initiate a logical connection and establish the security capabilities associated with that connection. This consists of two messages, the **client hello** and the **server hello**, as shown in Fig. 6.14.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

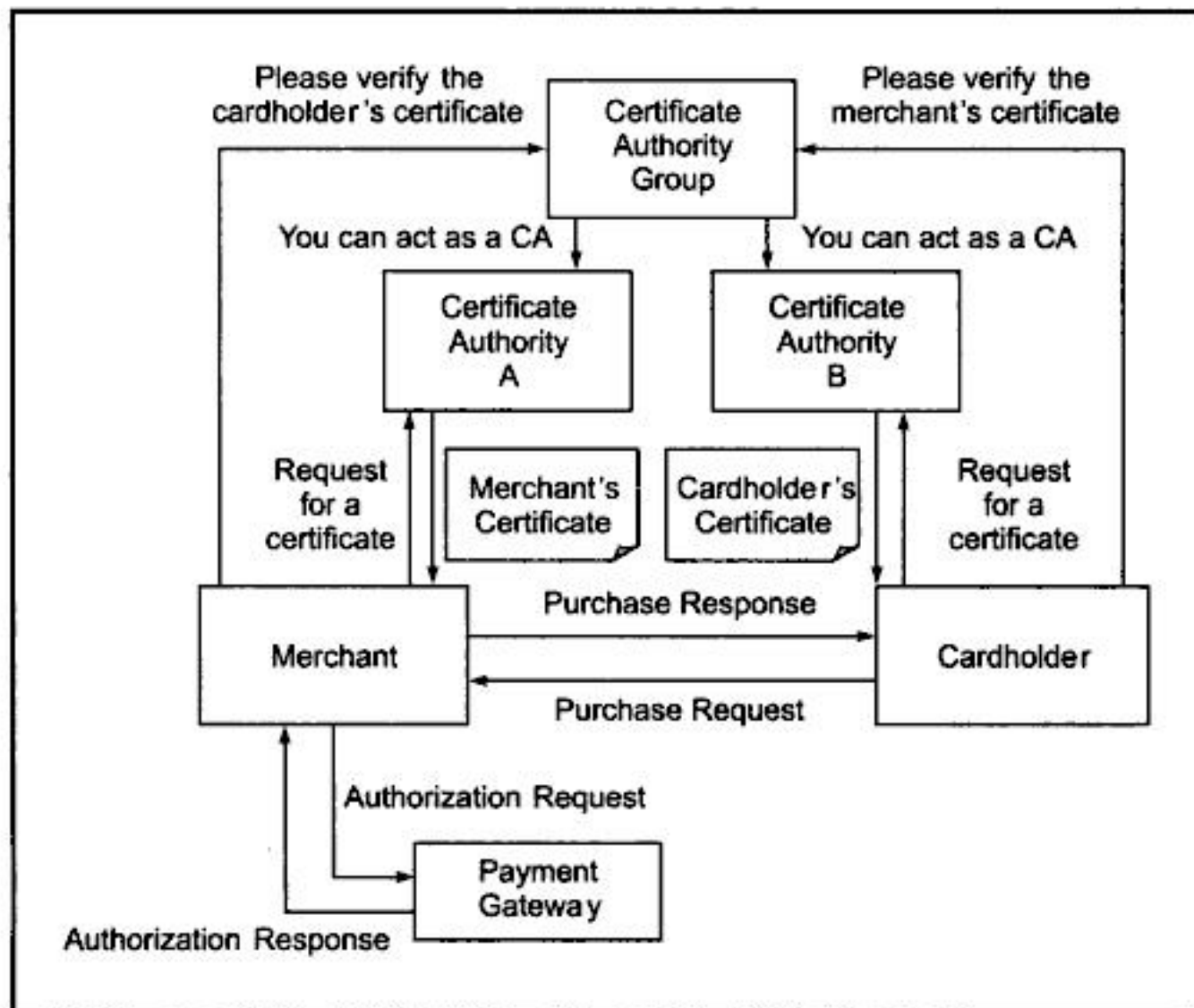




You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



┆ Fig. 6.38 The SET model

as many certificates as the number of different brands of credit cards that it accepts (e.g. one for MasterCard, one for Visa, one for Amex, and so on). Thus, when a customer receives a merchant certificate, it is also assured that the merchant is authorized to accept payments for that brand of credit card. This is similar to the boards displayed by real-life stores and restaurants that they accept certain credit cards.

As discussed, the transactions between a customer and merchant are for purchases, and those between the merchant and the payment authority are for authorization of payment. This is described in detail earlier.



## 6.8 SSL Versus SET

Having discussed SSL and SET in detail. Let us take a quick look at the differences between them, as shown in Table 6.5.

This table should give us an idea that SET is a standard that describes a very complex authentication mechanism that makes it almost impossible for either party to commit any sort of fraud. However, there is no such mechanism in SSL. In SSL, data is exchanged securely. However, the customer provides critical data such as credit card details to a merchant, and hopes that the customer does not misuse them. This is not possible in SET. Also, in the case of SSL, a merchant believes that the credit card really belongs to the customer and that he is not using a stolen card. In the case of SET, this is very unlikely and even if it happens, the merchant is safe, since the payment gateway has to ensure that the customer

**Table 6.5** *SSL versus SET*

<i>Issue</i>	<i>SSL</i>	<i>SET</i>
Main aim	Exchange of data in an encrypted form	E-commerce related payment mechanism
Certification	Two parties exchange certificates	All the involved parties must be certified by a trusted third party
Authentication	Mechanisms in place, but not very strong	Strong mechanisms for authenticating all the parties involved
Risk of merchant fraud	Possible, since customer gives financial data to merchant	Unlikely, since customer gives financial data to payment gateway
Risk of customer fraud	Possible, no mechanisms exist if a customer refuses to pay later	Customer has to digitally sign payment instructions
Action in case of customer fraud	Merchant is liable	Payment gateway is liable
Practical usage	High	Not much

is not committing fraud. The point is that SSL was created for exchanging secure messages over the Internet, whereas SET was specifically designed for secure e-commerce transactions involving online payment. So, these differences should not surprise anybody.

## 6.9 3-D Secure Protocol

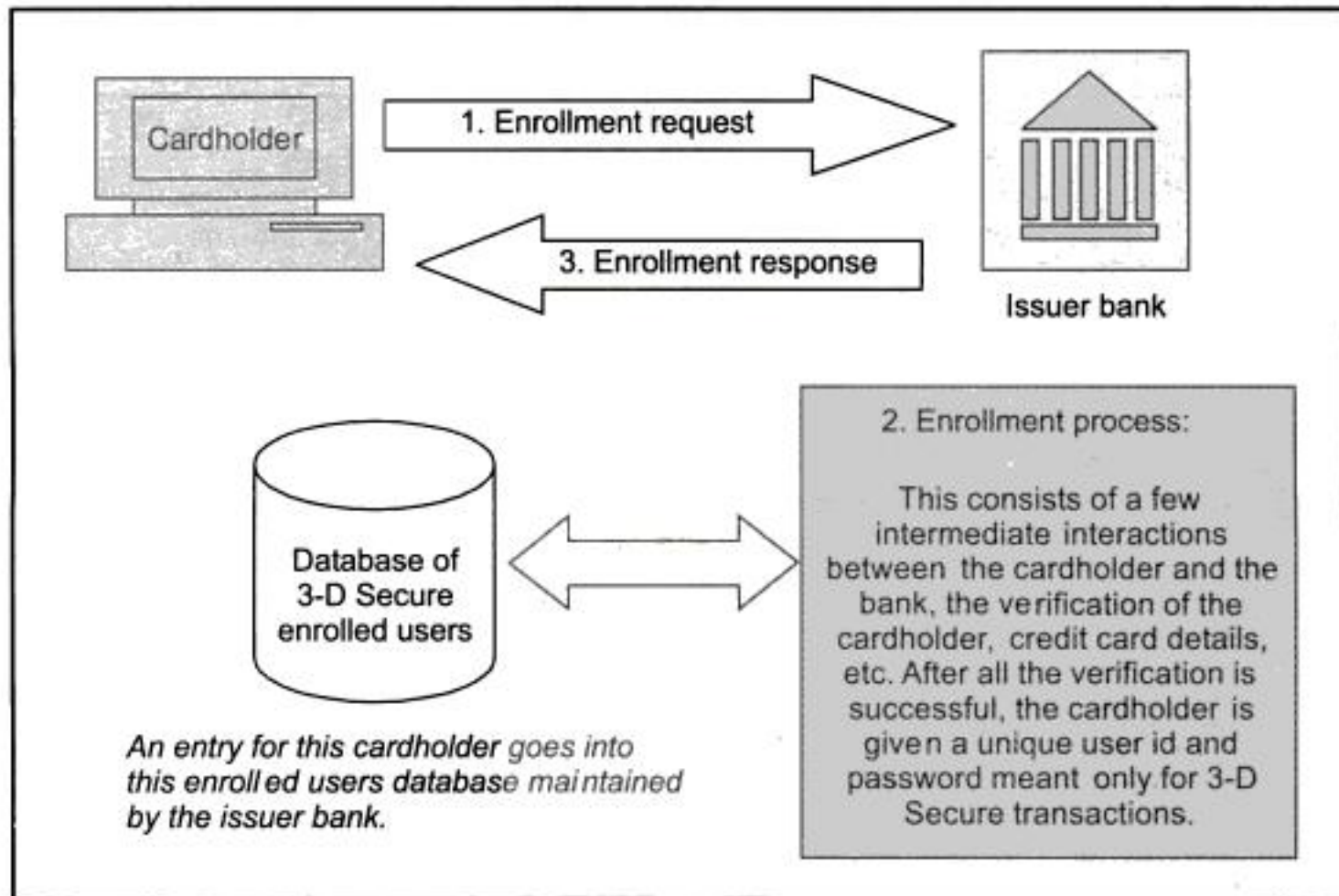
In spite of its advantages, SET has one limitation: it does not prevent a user from providing someone else's credit card number. The credit card number is protected from the merchant. However, how can one prevent a customer from using another person's credit card number? That is not achieved in SET. Consequently, a new protocol developed by Visa has emerged, called as **3-D Secure**.

The main difference between SET and 3-D Secure is that any cardholder who wishes to participate in a payment transaction involving the usage of the 3-D Secure protocol has to enroll on the issuer bank's *Enrollment Server*. That is, before a cardholder makes a card payment, she must enroll with the issuer bank's Enrollment server. This process is shown in Fig. 6.39.

At the time of an actual 3-D Secure transaction, when the merchant receives a payment instruction from the cardholder, the merchant forwards this request to the issuer bank through the Visa network. The issuer bank requires the cardholder to provide the user id and password that were created at the time of user enrollment process. The cardholder provides these details, which the issuer bank verifies against its 3-D Secure enrolled users database (against the stored card number). Only after the user is authenticated successfully that the issuer bank informs the merchant that it can accept the card payment instruction.

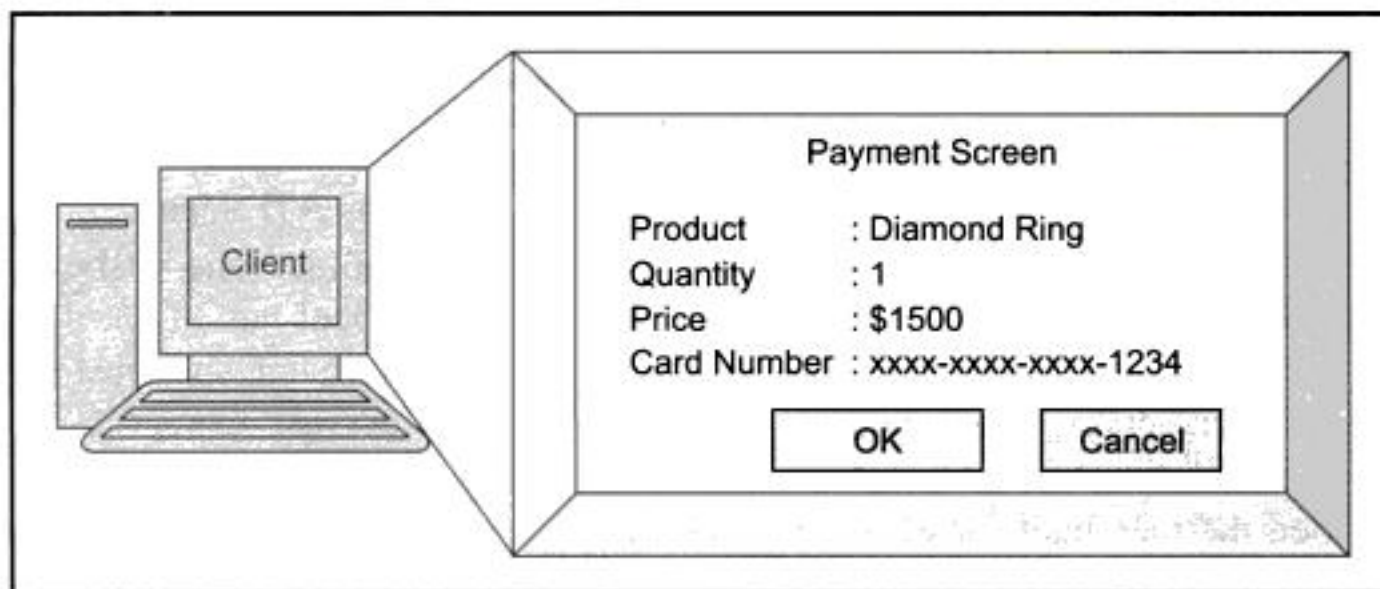
### 6.9.1 Protocol Overview

Let us understand how the 3-D Secure protocol works, step-by-step.



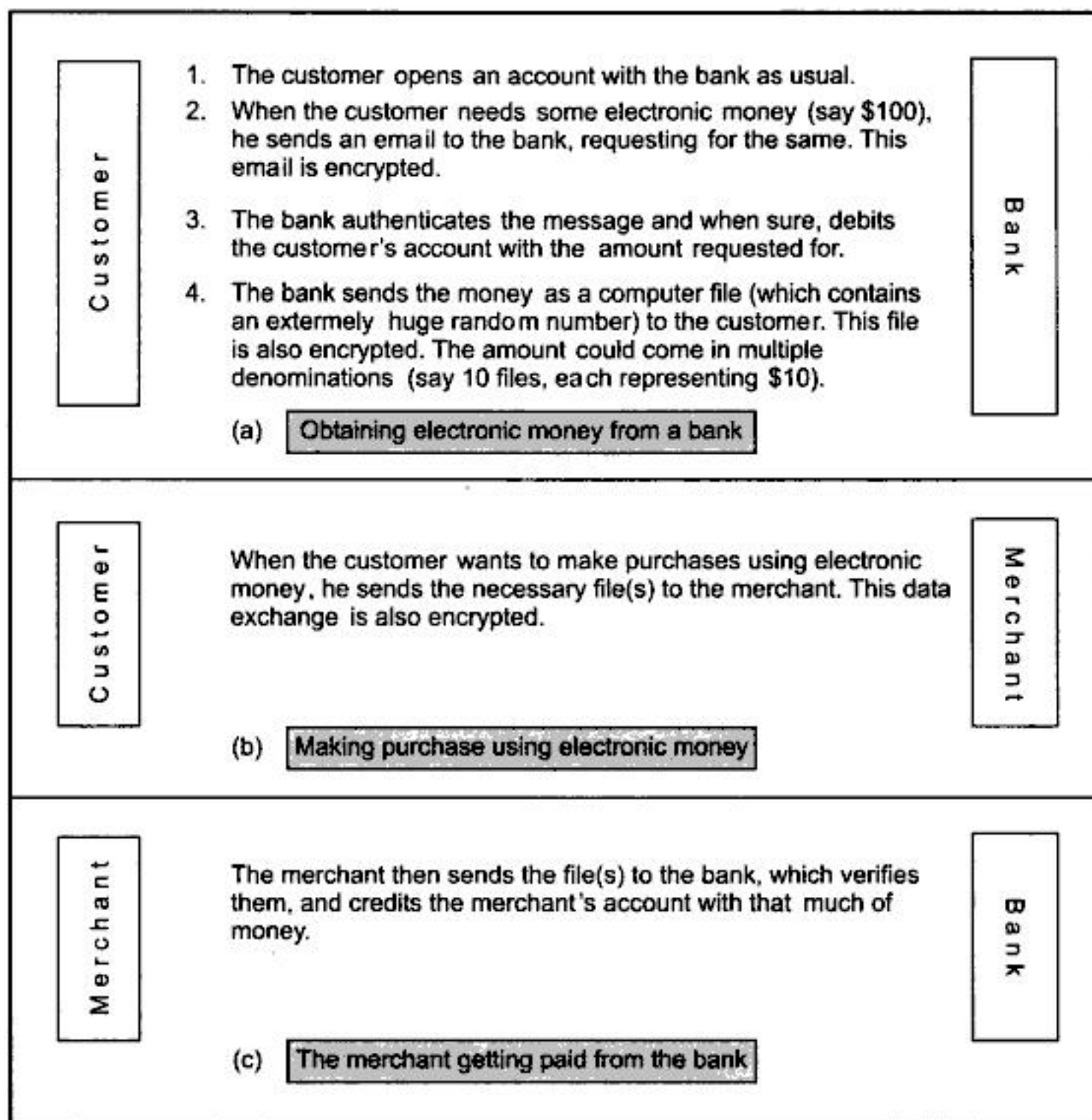
┆ Fig. 6.39 User enrollment

**Step 1** The user shops using the shopping cart on the merchant site, and decides to pay the amount. The user enters the credit card details for this purpose and clicks on the OK button, as shown in Fig. 6.40.



┆ Fig. 6.40 Step 1 in 3-D secure

**Step 2** When the user clicks on the *OK* button, the user will be redirected to the issuer bank's site. The bank site will popup a screen, prompting the user to enter the password provided by the issuer bank. This is shown in Fig. 6.41. The bank (issuer) authenticates the user by the mechanism selected by the user earlier. In this case, we consider a simple static id and password based mechanism. Newer trends involve sending a number to the user's mobile phone and asking the user to enter that number on the screen. However, that falls outside of the purview of the 3-D Secure protocol.



┆ Fig. 6.43 Model of electronic money

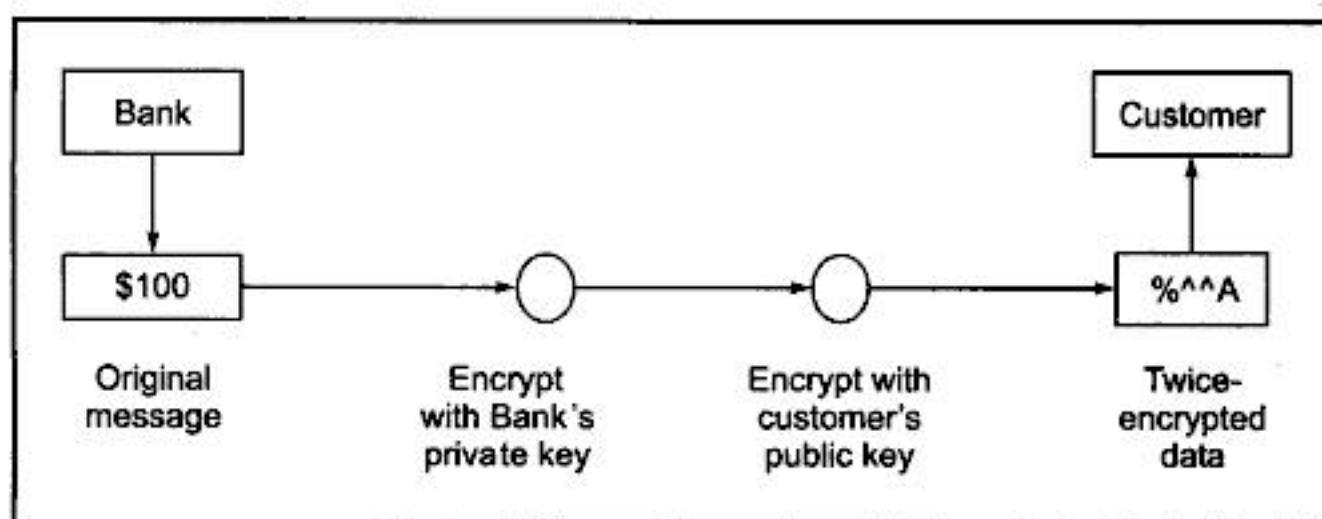
### 6.10.2 Security Mechanisms in Electronic Money

The security mechanisms in these procedures are similar to all the previous mechanisms described earlier. Let us study the process of the customer obtaining the money in the form of files from the bank. The same principles would apply in other transactions (e.g. a customer buying something from a merchant and then sending these files to him).

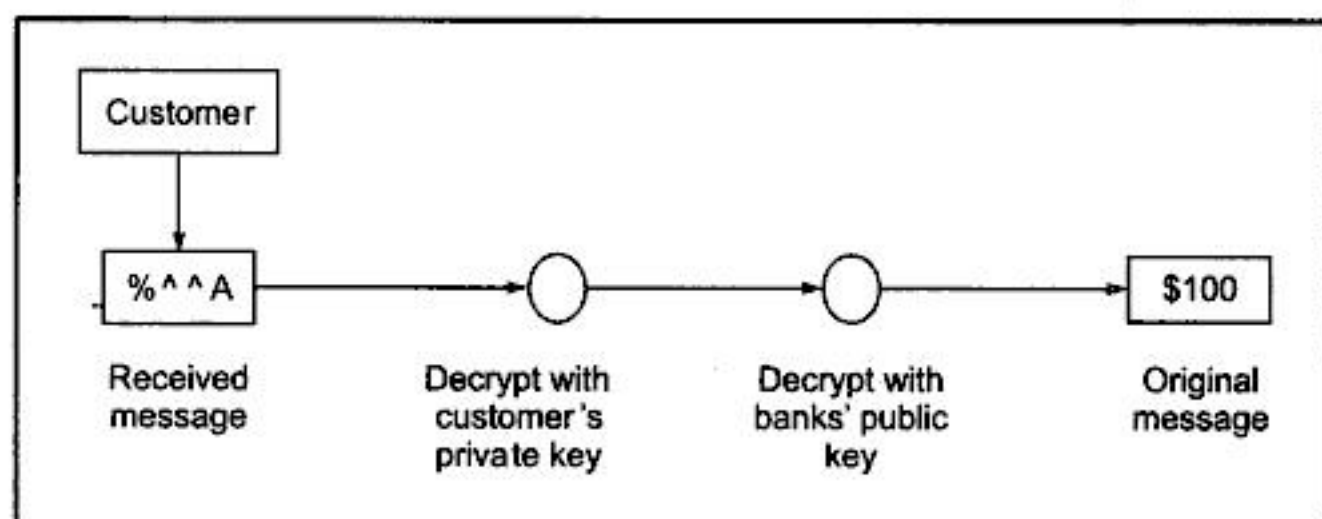
**Step 1** Bank sends the electronic money to the customer, as shown in Fig. 6.44.

As the figure shows, the bank first encrypts the original message with its own private key. It then encrypts this encrypted message further: this time with the customer's public key. Thus, the original message is encrypted twice. The bank sends this twice-encrypted message to the customer.

**Step 2** The customer receives the money and decrypts it, as shown in Fig. 6.45.



┆ Fig. 6.44 Bank sends electronic money to the customer after encrypting it twice



┆ Fig. 6.45 Customer decrypts the bank's message twice to get the electronic money

Here, the customer first decrypts the received message with its own private key. Further, it decrypts this once-decrypted message using the bank's public key. Thus, the customer gets the original message back (which is \$100). To ensure authentication, techniques of digital signatures and certificates may also be used in addition to these steps. We shall not describe those, as we have studied them earlier.

### 6.10.3 Types of Electronic Money

Electronic money can be classified in two ways. In the first classification, the types of electronic money are decided based on whether the electronic money is tracked or not. In this classification, electronic money can be of these two types: **identified electronic money** and **anonymous electronic money**. In the other method of classification, it is based on whether or not the transaction is real-time. In this classification, electronic money can be either **online electronic money** or **offline electronic money**. We shall study these types now.

**Classification Based on the Tracking of Money** This classification is based on whether the electronic money is tracked throughout its lifetime. Accordingly, it can be classified as follows.

**1. Identified electronic money** Identified electronic money more or less works like a credit card. The progress of the identified electronic money from the very first time it is issued by a bank to one of its customer, up to its final return to the bank can be easily tracked by the bank. As a result, the bank can precisely know how and when the money was spent by the customer. Consequently, the bank knows who is the original customer that had requested for this money and how he spent it. How is this



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



This process is shown in Fig. 6.47.

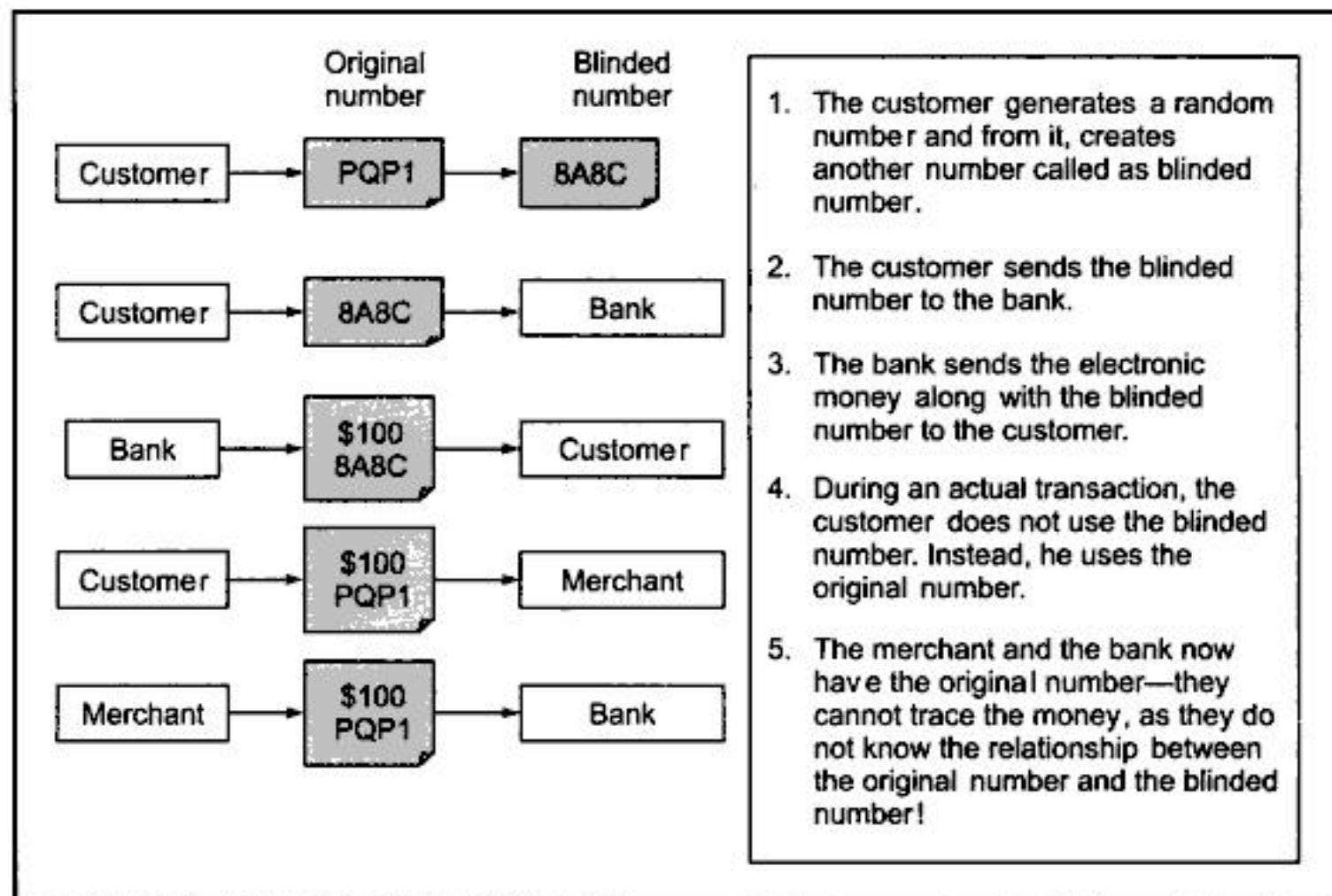


Fig. 6.47 Steps involved in anonymous electronic money

In the case of identified electronic money, the chances of a customer trying to spend the same money more than once can be easily caught or prevented. This is possible because the bank maintains a list of the issued and spent serial numbers. Therefore, it can catch attempts of spending the same piece of electronic money more than once.

#### 6.10.4 Classification Based on the Involvement of the Bank in the Transaction

Based on involvement (or otherwise) of the bank in the actual transaction, electronic money can be further classified into two categories: online electronic money and offline electronic money.

##### (a) Online electronic money

In this type, the bank must actively participate in the transaction between the customer and the merchant. That is, before the purchase transaction of the customer can complete, the merchant would confirm from the bank in real time as to whether the electronic money offered by the customer is acceptable (e.g. ensuring that it is not already spent, or that the serial number for it is valid).

##### (b) Offline electronic money

In this type, the bank does not participate in the transaction between the customer and the merchant. That is, the customer purchases something from the merchant and offers to pay by electronic money. The merchant accepts the electronic money, but does not validate it online. The merchant might collect a group of such electronic money transactions and process them together at a fixed time every day.

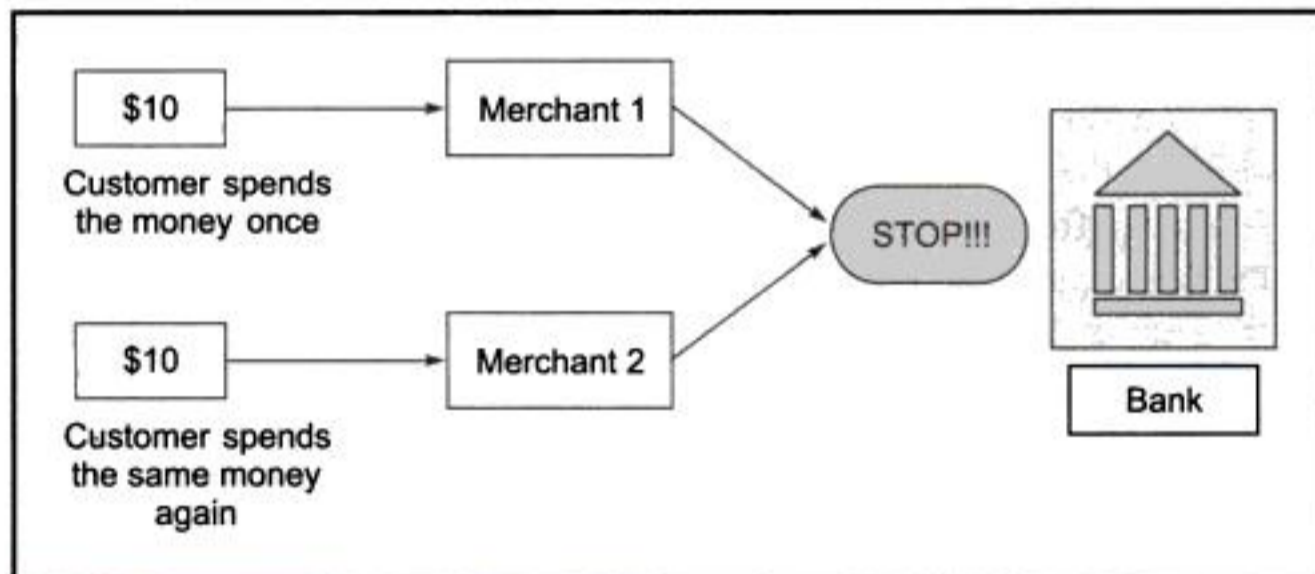
### 6.10.5 The Double Spending Problem

Now, if we combine the two ways of classifying electronic money, we have four possibilities:

1. Identified Online electronic money
2. Identified Offline electronic money
3. Anonymous Online electronic money
4. Anonymous Offline electronic money

Of the four, the last type can create the **double spending problem**. More specifically, a customer could arrange for anonymous electronic money by using the blinded money concept. Later on, he could spend it offline more than once in quick succession (say in the same hour) with two different merchants. Since the bank is not involved in any of the two online transactions, the fact that the same piece of money is being spent cannot be prevented. Moreover, when it is realized that the same piece of money is spent more than once (when both merchants send their daily transaction lists to the bank), the bank cannot determine which customer spent it more than once, because of the blinding factor (recall our discussion of anonymous electronic money). Consequently, *anonymous offline electronic money* is of little practical use.

Double spending problem can happen in case of *identified offline electronic money* as well. However, upon detection, the customer under question can be easily tracked from the serial numbers of the electronic money. This is shown in Fig. 6.48.



— Fig. 6.48 Detection of double spending problem

However, this detection is not possible in case of *anonymous offline electronic money*. Of course, in case of either of the online electronic money transactions, double spending problem is simply not possible, since the bank is a part of the transaction between the customer and the merchant.



## 6.11 Email Security

### 6.11.1 Introduction

Electronic mail (email) is perhaps the most widely used application on the Internet. Using email, an Internet user can send a message (and these days, pictures, video, sound, etc) to other Internet user(s). Consequently, the security of email messages has become an extremely important issue. Before we study email security, let us quickly review the email technology in brief.

```

S: 220 hotmail.com Simple Mail Transfer Service Ready
C: HELO yahoo.com
S: 250 hotmail.com

C: MAIL FROM: <Atul@yahoo.com>
S: 250 OK

C: RCPT TO: <Ana@hotmail.com>
S: 250 OK

C: RCPT TO: <Jui@hotmail.com>
S: 250 OK

C: DATA
S: 354 Start mail input; end with <CR><LF><LF>
C: ... actual contents of the message ...
C: .....
C: .....
C: <CR><LF><LF>
S: 250 OK

C: QUIT
S: 221 hotmail.com Service closing transmission channel

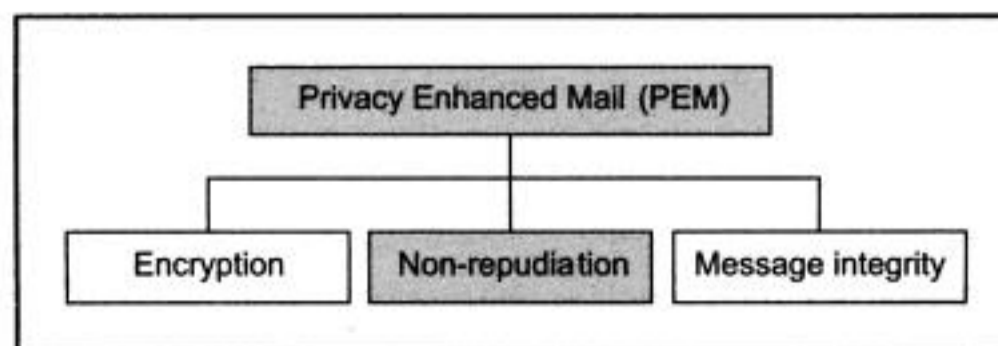
```

┆ Fig. 6.51 Example of an email message using SMTP protocol

Having discussed the basic concepts of email communication let us now study the three main email security protocols: **Privacy Enhanced Mail (PEM)**, **Pretty Good Privacy (PGP)** and **Secure MIME (S/MIME)**.

### 6.11.2 Privacy Enhanced Mail (PEM)

**Introduction** The **Privacy Enhanced Mail (PEM)** is an email security standard adopted by the Internet Architecture Board (IAB) to provide secure electronic mail communication over the Internet. PEM was initially developed by the Internet Research Task Force (IRTF) and Privacy Security Research Group (PSRG). They then handed over the PEM standard to the Internet Engineering Task Force (IETF) PEM Working Group. PEM is described in four specification documents, which are RFC numbers 1421 to 1424. PEM supports the three main cryptographic functions of encryption, non-repudiation, and message integrity, as shown in Fig. 6.52.



┆ Fig. 6.52 Security features offered by PEM



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

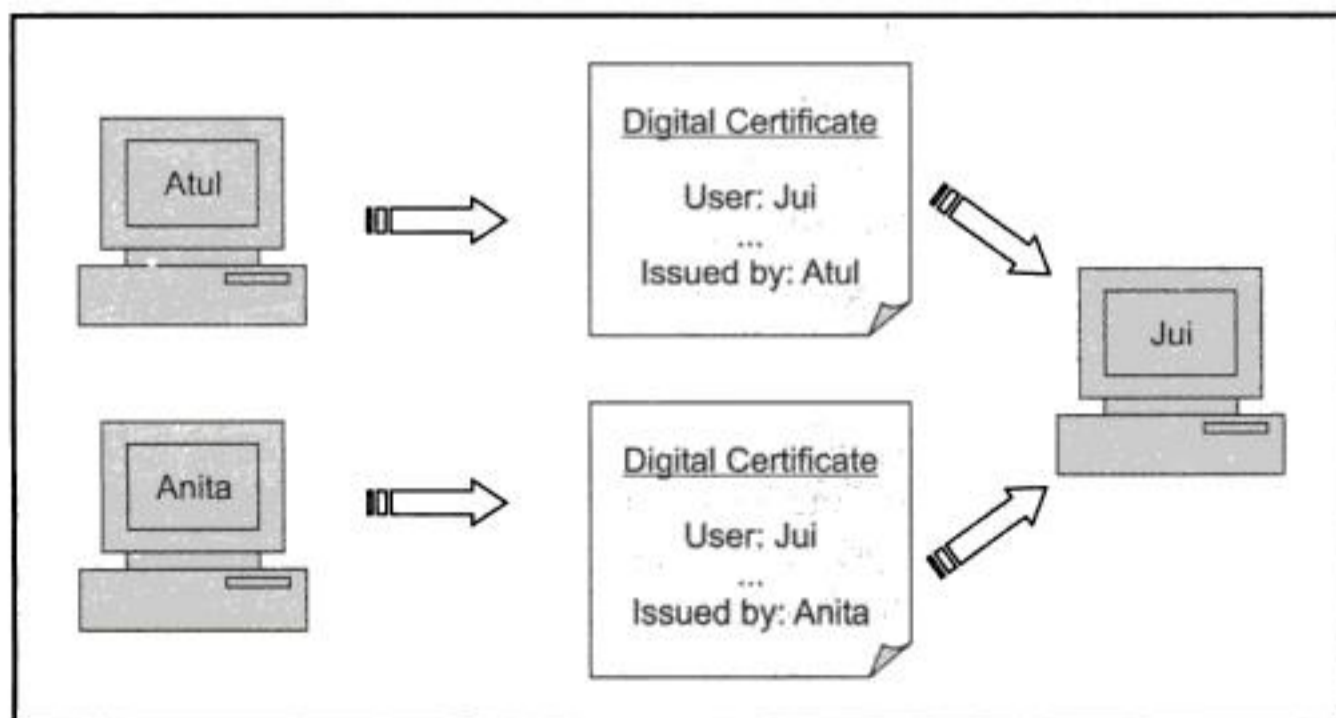


- (e) Alice encrypts the digital signature with the one-time symmetric key (using IDEA or DES-3 algorithm).
  - (f) Alice sends the output of Steps (d) and (e) above to the receiver. What would the receiver need to do? This is explained as follows:
2. Now suppose that Alice has received a message from one of the other users in the system.
    - (a) Alice uses her private key to obtain the one-time symmetric key created by the sender. Refer to steps (b) and (c) in the earlier explanation if you do not understand this.
    - (b) Alice uses the one-time symmetric key to decrypt the message. Refer to steps (b) and (d) in the earlier explanation if you do not understand this.
    - (c) Alice computes a message digest of the original message (say MD1).
    - (d) Alice now uses this one-time symmetric key to obtain the original digital signature. Refer to steps (b) and (e) in the earlier explanation if you do not understand this.
    - (e) Alice uses the sender's public key from the key ring shown in the right side of the diagram to decrypt the digital signature and gets back the original message digest (say MD2).
    - (f) Alice compares message digests MD1 and MD2. If they match, Alice is sure about the message integrity and authentication of the message sender.

**PGP Certificates** In order to trust the public key of a user, we need to have that user's digital certificate. PGP can use certificates issued by a CA, or can use its own certificate system.

As explained in our discussion of digital certificates, in X.509, there is a root CA, who issues certificates to the second level CAs. The second level CAs can issue certificates to the third level CAs and so on. This can continue up to the required number of levels. At the lowest level, the last CA issues certificates to the end users.

In PGP, things work differently. There is no CA. Any one can sign a certificate belonging to anyone else in the ring. Atul can sign the certificate for Ana, Jui, Harsh and so on. There is no hierarchy of trust, or a tree-like structure. This creates a situation where a user can have certificates issued by different users. For example, Jui may have a certificate signed by Atul, and another one by Anita. This is shown in Fig. 6.64. Hence, if Harsh wants to verify Jui's certificate, he has two paths: Jui -> Atul, and



┆ Fig. 6.64 Anyone can issue certificates to anyone else in PGP

2. Ravi issues a certificate to Uday (with public key K3). Mahesh stores the public key and certificate of Uday in his ring of public keys with *certificate trust* level equal to *partial*.
3. Amol issues two certificates: one to Uday (with public key K3), and another to Parag (with public key K4). Mahesh stores the public keys and certificates of Uday and Parag in his ring of public keys with *certificate trust* level equal to *partial*. Note that Mahesh now has two certificates for Uday, one issued by Ravi, and the other issued by Amol, both with *partial* level of *certificate trust*.
4. Amit issues a certificate to Pramod (with public key K4). Mahesh stores the public key and certificate of Pramod in his ring of public keys with *certificate trust* level equal to *none*. Mahesh can also discard this certificate.

**Key Legitimacy** The objectives behind introducer trust and certificate trust is to decide whether to trust the public key of a user. In PGP terms, this is called as **key legitimacy**. Mahesh needs to know how legitimate are the public keys of Amrita, Pallavi, Uday, Parag, Pramod and so on.

PGP defines the following simple rule to decide the key legitimacy: *The level of key legitimacy for a user is the weighted trust level for that user.* For instance, suppose that we have assigned certain weights to certificate trust levels, as shown in Fig. 6.67.

Weight	Meaning
0	No trust
$\frac{1}{2}$	Partial trust
1	Complete or full trust

Fig. 6.67 Assigning weights to certificate trust levels

In this situation, in order to trust a public key (i.e. certificate) of any other user, Mahesh needs one fully trusted certificate or two partially trusted certificates. Thus, Mahesh can fully trust Amrita and Pallavi based on the certificates they had received from Naren. Mahesh can also trust Uday, based on the two partially trusted certificates that Uday had received from Ravi and Amol.

Interestingly, the legitimacy of a public key belonging to an entity has nothing to do with the trust level of that person. For instance, Naren may be trusting Amit. Hence, Naren can encrypt a message with the public key derived from Amit's certificate and can send the encrypted message to Amit. However, Mahesh will continue to reject certificates issued by Amit, since he does not trust Amit.

**Web of Trust** The earlier discussion leads to a potential problem. What happens if nobody creates a certificate for a fully or partially trusted entity? In our example, on what basis would we trust Naren's public key if no one has created a certificate for Naren? To resolve this problem, several schemes are possible in PGP, as outlined below.

- (a) Mahesh can physically obtain the public key of Naren by meeting in person and getting the key on a piece of paper or as a disk file.
- (b) This can be done telephonically as well.
- (c) Naren can email his public key to Mahesh. Both Naren and Mahesh compute a message digest of this key. If MD5 is used, the result is a 16-byte digest. If SHA-1 is used, the result is a 20-byte digest. In hexadecimal, the digest becomes a 32-digit value in MD5, and a 40-digit value in SHA-1. This is displayed as 8 groups of 4-digit values in MD5, or 10 groups of 4-digit values in SHA-1, and is called as **fingerprint**. Before Mahesh adds the public key of Naren to his ring, he can call up Naren to tell him what fingerprint value he has obtained to cross-check with the fingerprint value that is separately obtained by Naren. This ensures that the public key value is not changed in the email transit. To make matters better, PGP assigns a unique English word to a 4-digit hexadecimal number group, so that instead of speaking out the hexadecimal string of numbers,

users can speak out normal English words, as defined by PGP. For example, PGP may have assigned a word *India* to a hexadecimal pattern of *4A0B*, etc.

(d) Mahesh can, of course, obtain Naren's public key from a CA.

Regardless of the mechanism, eventually this process of obtaining keys of other users and sending our own to others creates what is called as a **web of trust** between groups of people. This keeps the public key ring getting bigger and bigger, and helps secure the email communication.

Whenever a user needs to revoke her public key (because of loss of private key, etc), she needs to send a *key revocation certificate* to the other users. This certificate is self-signed by the user with her private key.

#### 6.11.4 Secure Multipurpose Internet Mail Extensions (S/MIME)

**Introduction** The traditional email systems using the SMTP protocol (based on RFC 822) are text-based, which means that a person can compose a text message using an editor and then send it over the Internet to another recipient. However, in the modern era, exchanging only text messages is not quite sufficient. People want to exchange multimedia files, documents in various arbitrary formats, etc. To cater to these needs, the **Multipurpose Internet Mail Extensions (MIME)** system extends the basic email system by permitting users to send binary files using the basic email system. MIME is defined in RFCs 2045 to 2049.

Knowing why SMTP cannot work with non-text data is important. SMTP uses the 7-bit ASCII representation for characters of an email message. 7-bit ASCII cannot represent special characters above the ASCII value of 127. SMTP cannot also send binary data.

A MIME email message contains a normal Internet text message along with some special headers and formatted sections of text. Each such section can hold an ASCII-encoded portion of data. Each section starts with an explanation as to how the data that follows should be interpreted/decoded at the recipient's end. The recipient's email system uses this explanation to decode the data.

Let us consider a simple example containing MIME header. Figure 6.68 shows an email message (after the sender has composed the email message and has attached a graphics file having .GIF extension). The figure shows the email message as it actually travels to the recipient. The Content-Type MIME header shows the fact that the sender has attached a .GIF file to this message. The actual image would be sent as a part of the message after this, and would appear gibberish when viewed in the text form (because it would be the binary representation of the image). However, the recipient's email system shall recognize that this is a .GIF file, and as such, it should invoke an appropriate program that can read, interpret and display contents of a .GIF file.

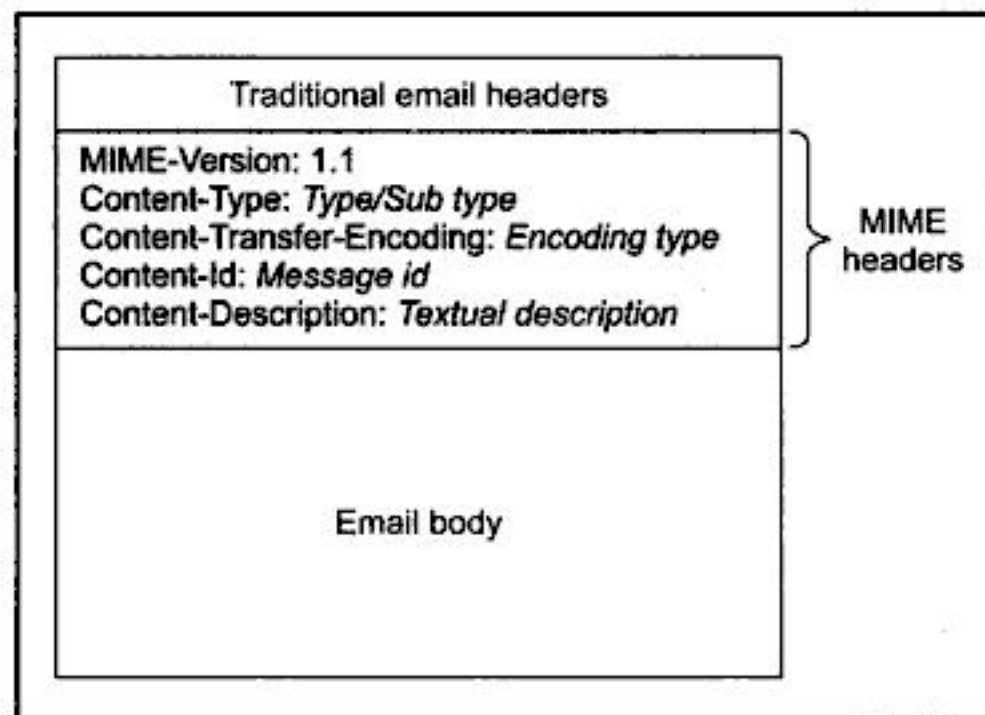
```
From: Atul Kahate <akahate@indiatimes.com>
To: Anita Kahate <akahate@yahoo.com>
Subject: Cover image for the book
MIME-Version: 1.0
Content-Type: image/gif

<Actual image data in the binary form such as R019a0asdjas0 ...>
```

— Fig. 6.68 MIME Extensions to an email message

When we enhance the basic MIME system to provide for security features, it is called as **Secure Multipurpose Internet Mail Extensions (S/MIME)**.

**MIME Overview** We have seen that the email system provides for email headers such as *From*, *To*, *Date*, *Subject*, etc. The MIME specification adds five new headers to the email system, which describe information about the body of the message (we have already seen two MIME headers in our example just now). Thus, when MIME is in use, the email message looks as shown in Fig. 6.69.



┆ Fig. 6.69 MIME headers in an email message

These headers are described as follows:

- o **MIME-Version:** This contains the MIME version number. Currently, it has a value of 1.1. This field is reserved for the future use, when newer versions of MIME are expected to emerge. This field indicates that the message conforms to RFCs 2045 and 2046.
- o **Content-Type:** Describes the data contained in the body of the message. The details provided are sufficient so that the receiver email system can deal with the received email message in an appropriate manner. The contents are specified as:

Type/Sub-type

MIME specifies 7 content types, and 15 content sub-types. Table 6.8 lists these types and sub-types.

- o **Content-Transfer-Encoding:** Specifies the type of transformation that has been used to represent the body of the message. In other words, the method used to encode the messages into zeroes and ones is defined here. There are five content encoding methods, as shown in Table 6.9.
- o **Content-ID:** Identifies the MIME entities uniquely with reference to multiple contexts.
- o **Content-Description:** Used when the body is not readable (e.g. video).

**S/MIME Functionality** In terms of the general functionality, S/MIME is quite similar to PGP. Like PGP, S/MIME provides for digital signatures and encryption of email messages. More specifically, S/MIME offers the functionalities as depicted in Table 6.10.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

8. Credit card details are not available to the \_\_\_\_\_ in SSL.
  - (a) merchant
  - (b) customer
  - (c) payment gateway
  - (d) issuer
9. Electronic money is made up of \_\_\_\_\_ in physical form.
  - (a) floppy disks
  - (b) computer files
  - (c) hard disks
  - (d) credit card
10. \_\_\_\_\_ money can create the double spending problem.
  - (a) Identified, online
  - (b) Identified, offline
  - (c) Anonymous, online
  - (d) Anonymous, offline
11. PEM allows for \_\_\_\_\_ security options.
  - (a) 2
  - (b) 3
  - (c) 4
  - (d) 5
12. The \_\_\_\_\_ protocol needs to identify the content type before email can be transmitted.
  - (a) PEM
  - (b) PGP
  - (c) SMTP
  - (d) MIME
13. In \_\_\_\_\_, we have the concept of key rings.
  - (a) PEM
  - (b) PGP
  - (c) SMTP
  - (d) MIME
14. In \_\_\_\_\_, the user needs to authenticate before using a credit card in an electronic transaction.
  - (a) SET
  - (b) SSL
  - (c) 3-D secure
  - (d) WTLS
15. The security layer in WAP is between the \_\_\_\_\_ layer and the \_\_\_\_\_ layer.
  - (a) transaction, transport
  - (b) application, transport
  - (c) transport, physical
  - (d) session, transport



## EXERCISES

1. Why is the SSL layer positioned between the application layer and the transport layer?
2. What is the purpose of the SSL alert protocol?
3. Explain the SSL handshake protocol.
4. How is SHTTP different from SSL?
5. What is the significance of the time stamping protocol?
6. Which are the key participants in SET?
7. How does SET protect payment information from the merchant?
8. Outline the broad level steps in SET.
9. How is 3-D Secure different from SET?
10. What is electronic money?
11. Why is anonymous offline electronic money dangerous? Discuss the double spending problem.
12. Mention the broad level steps in PEM and PGP.
13. Explain the concept of key rings in PGP.

14. What is the security concern in WAP?
15. How does GSM security work?



## DESIGN/PROGRAMMING EXERCISES

1. Many real-life algorithms, including RSA and SSL; use the principle of GCD. Find out why the GCD of any two consecutive integers,  $n$  and  $n+1$  is always 1. (Hint: Refer to Appendix A).
2. Find out the GCD of two numbers: 42120 and 46510.
3. Write a Java implementation of the GCD method provided in Appendix A (which provides a similar implementation in C).
4. Write a C program to test whether an inputted number is prime or not.
5. Achieve the same objective as above, using Java.
6. Assume a 24-bit input as 101010111100000101100110 and transform it into its Base-64 equivalent, using the algorithm shown in this chapter.
7. Write a C program to perform Base-64 encoding on a 24-bit input.
8. Think about what will happen if we do not have Base-64 encoding mechanism.
9. Can we implement the concept of key rings on the Internet (e.g. when we purchase something over the Internet)? Why?
10. Consider the following text:

Welcome to the world of Security. The world of Security is full of interesting problems and solutions.

How would the Lempel-Ziv algorithm compress this text? Consider a conceptual view, where we want to compress (replace) the words *to*, *the*, *of* and *security*.
11. Investigate more about the differences between GSM and 3G technologies from a security standpoint.
12. If we implement both WTLS and GSM/3G security, would it offer any extra security? Would one of these be redundant? Why?
13. SSL talks about security at the transport layer. What if we want to enforce security at the lower layers?
14. Investigate how to use SSL in Java and .NET. Write an SSL client and server in these technologies.
15. Apache has implemented an open source library of SSL components. Investigate more and implement SSL using it.

# User Authentication and Kerberos



## 7.1 Introduction

One of the key aspects of cryptography and network/Internet security is authentication. Authentication helps establish trust by identifying who a particular user/system is. Authentication ensures that the claimant is really what he/she claims to be. This chapter discusses the various aspects of authentication mechanisms.

There are many ways to authenticate a user. Traditionally, user ids and passwords have been used. But there are many security concerns in this mechanism. Passwords can travel in clear text or can be stored in clear text on the server, both of which are dangerous propositions. Modern password-based authentication techniques use alternatives as encrypting passwords, or using something derived from the passwords in order to protect them.

Authentication tokens add randomness to the password-based mechanism and make it far more secure. This mechanism requires the user to possess the tokens. Authentication tokens are quite popular in applications that demand high security.

Certificate-based authentication has emerged as a modern authentication mechanism, thanks to the emergence of the PKI technology. This is also quite strong, if implanted correctly. Smart cards can also be used in conjunction with this technology. Smart cards facilitate cryptographic operations inside the card, making the whole process a lot more secure and reliable.

Biometrics is also getting a lot of attention these days, and is based on human biological characteristics. However, it has still not matured completely.

This chapter examines all these mechanisms of authentication in great detail. It discusses the advantages and drawbacks of each one of them. The chapter then concludes with the coverage of Kerberos, a single sign on mechanism implemented in many real-life systems.

## 7.2 Authentication Basics

*Who are you?* is a question that we ask everyday and get asked. It has a lot of importance and significance in the world of cryptography. As we studied earlier, the whole concept of authentication is





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

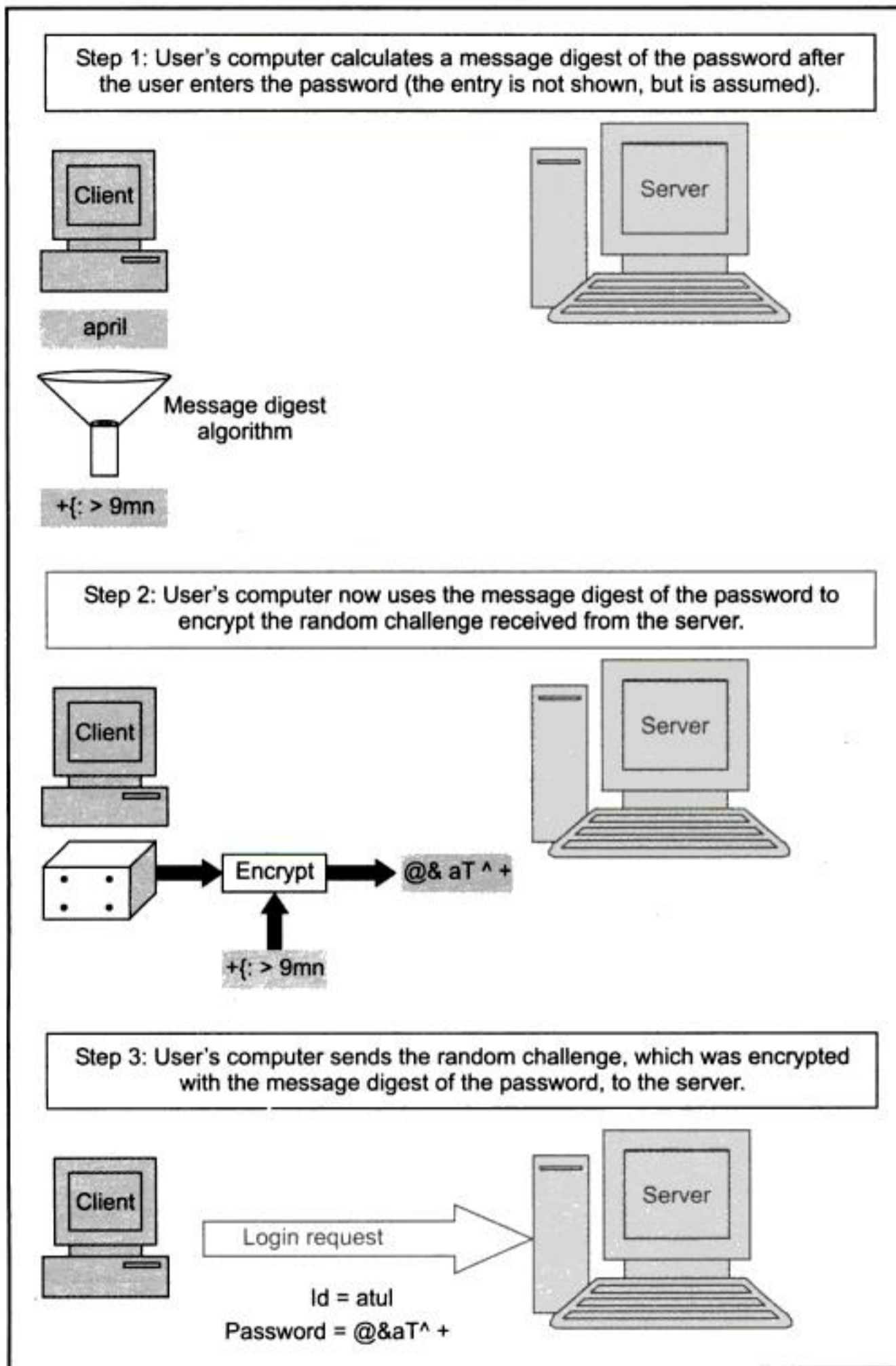


Fig. 7.12 User sends the user id and encrypted random challenge to the server

Note that the random challenge is different every time. Therefore, the random challenge encrypted with the message digest of the password would also be different every time. Therefore, an attacker attempting a *replay attack* is quite unlikely to succeed now. This is the basis for many real-life authentication mechanisms, including Microsoft Windows NT 4.0. Windows NT 4.0 uses the MD4

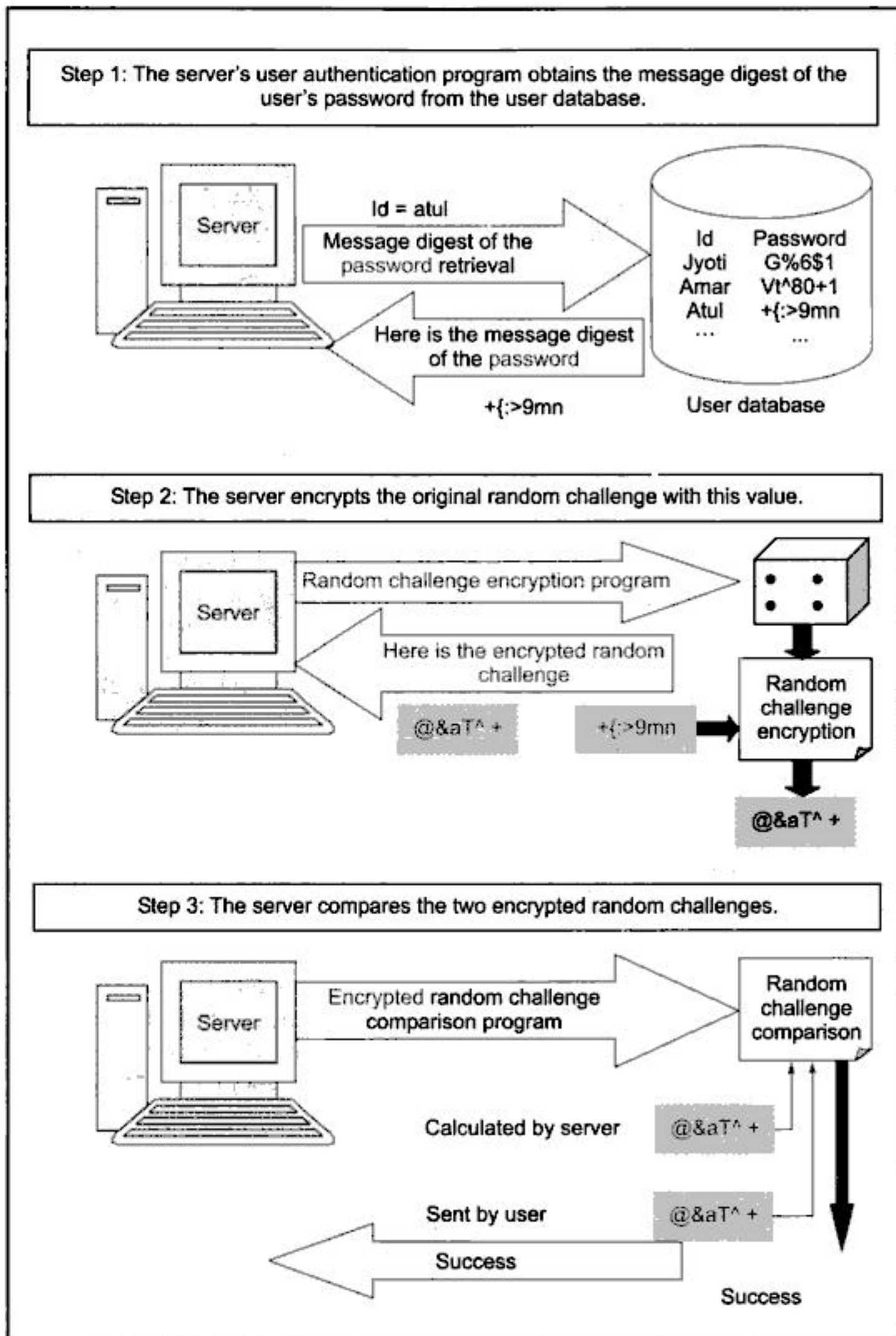
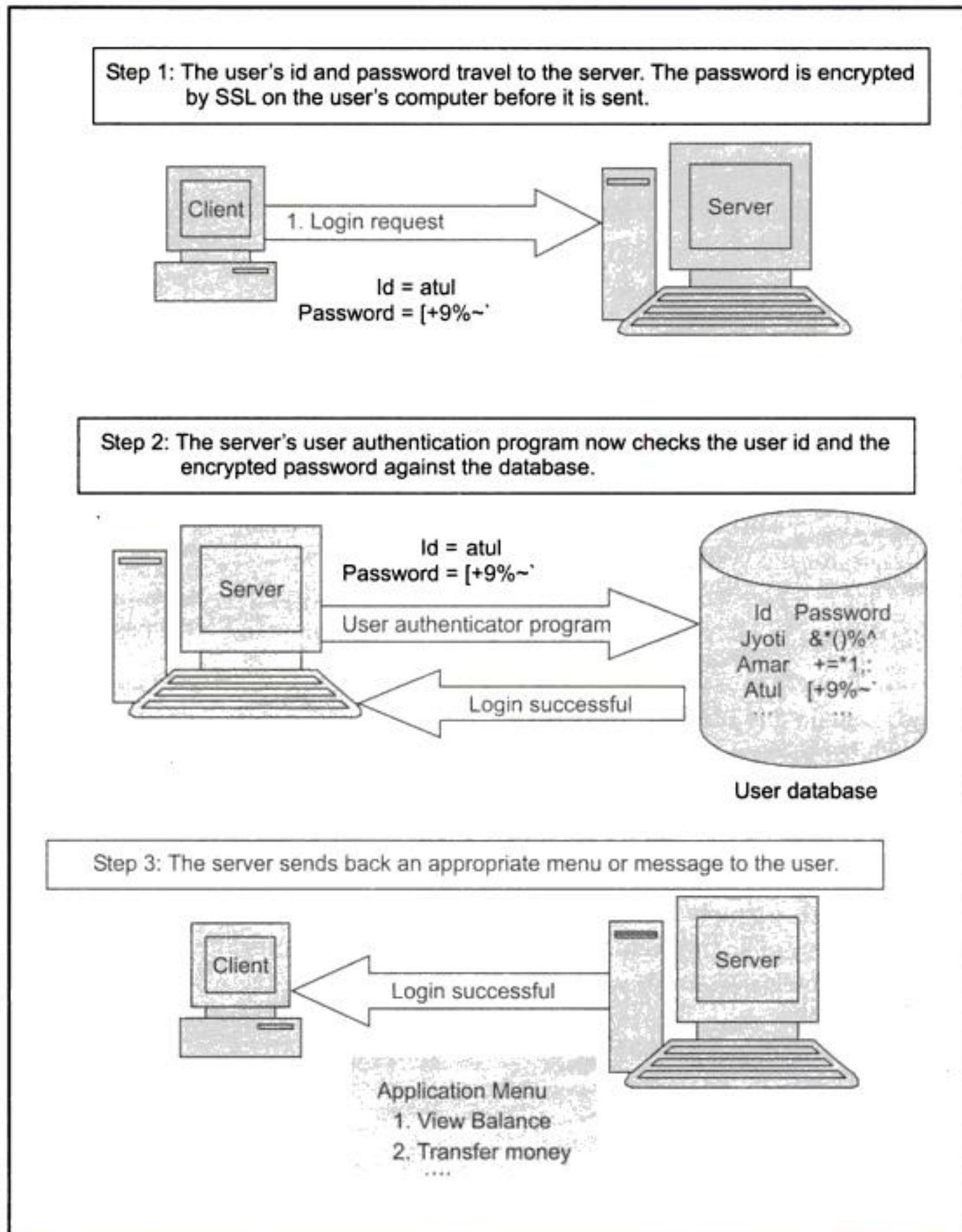


Fig. 7.13 Server compares the two encrypted random challenges





┆ Fig. 7.15 *Encrypting password on the client's computer and also in the database*

perform any transmission). Therefore, the encrypted password in the database would not actually be the same as the encrypted password coming from the user's computer. However, the main idea here is that both are encrypted passwords – neither of them is in clear text. That is the purpose of this illustration. The fact that the encrypted versions of these two passwords may not be the same and that the server-side application logic would perform the necessary conversions between the two for verification is a minor technical variation, which we shall ignore.

### 7.3.4 The Problems with Passwords

As we mentioned, a major misconception is that passwords are the simplest and the cheapest form of authentication mechanisms. An end user might actually be right in having this perspective. However, from an application or system administration point of view, this is quite incorrect.

Typically, an organization has a number of applications, networks, shared resources and intranets. Worse yet, these applications have varying needs of security measures and they grow over a period of time. Therefore, each such resource demands its own user id and password. This means that an end user has to remember and correctly use many user ids and passwords. To overcome the troubles associated with this, most users either keep the same password for all the resources, or record their passwords at some place. These places can be really weird. For instance, many users keep the passwords in their electronic diaries, or paper diaries, or cupboards, below the keyboard, or sometimes even stick to their monitor! Either way, this creates great concerns for the security of passwords, and therefore, the access of the resources.

Password maintenance is a very big concern for system administrators. A study shows that system administrators spend about 40% of their time creating, resetting or changing user passwords! This can truly be a nightmare for the system administrators.

Organizations specify **password policies**, which mandate the structure of passwords. For instance, an organization policy could have some of the following policies governing the passwords of its users:

- ✓ The password length must be at least 8 characters.
- ✓ It must not contain any blanks.
- ✓ There must be at least one lower case alphabet, one upper case alphabet, one digit and one special character in the password.
- ✓ The password must begin with an alphabet.

As we can see, this (like a salt in PBE, as discussed earlier) can be a significant deterrent to dictionary attacks, whereby attackers simply take normal words (from a dictionary) and try them as passwords. However, this creates a problem of remembering cryptic passwords for the end users. Therefore, end users resort to writing their passwords somewhere, which can defeat the whole purpose of a password policy!

In a nutshell, there are no easy solutions here!



## 7.4 Authentication Tokens

### 7.4.1 Introduction

An **authentication token** is an extremely useful alternative to a password. An authentication token is a small device that generates a new random value every time it is used. This random value becomes the basis for authentication. The small devices are typically of the size of small key chains, calculators or credit cards. Usually an authentication token has the following features:

- Processor
- Liquid Crystal Display (LCD) for displaying outputs
- Battery
- (Optionally) a small keypad for entering information
- (Optionally) a real-time clock



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



The server now needs to verify the user's signature. For this purpose, the server consults the user database to obtain the user's public key (since that can alone verify the user's signature). It then uses this public key to decrypt (also called as de-sign) the signed random challenge received from the user. After this, it compares this decrypted random challenge with its original random challenge. This is shown in Fig. 7.34.

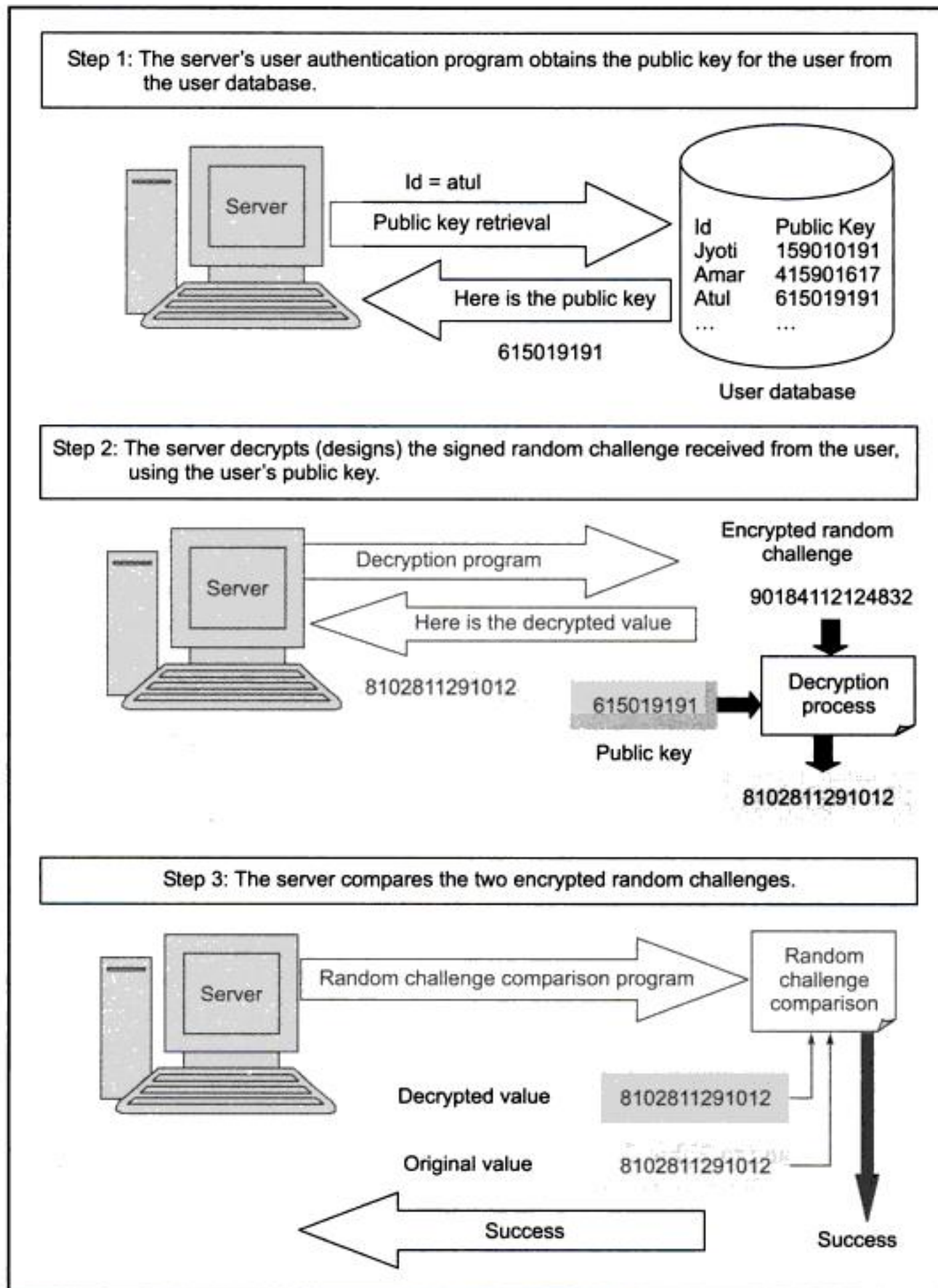
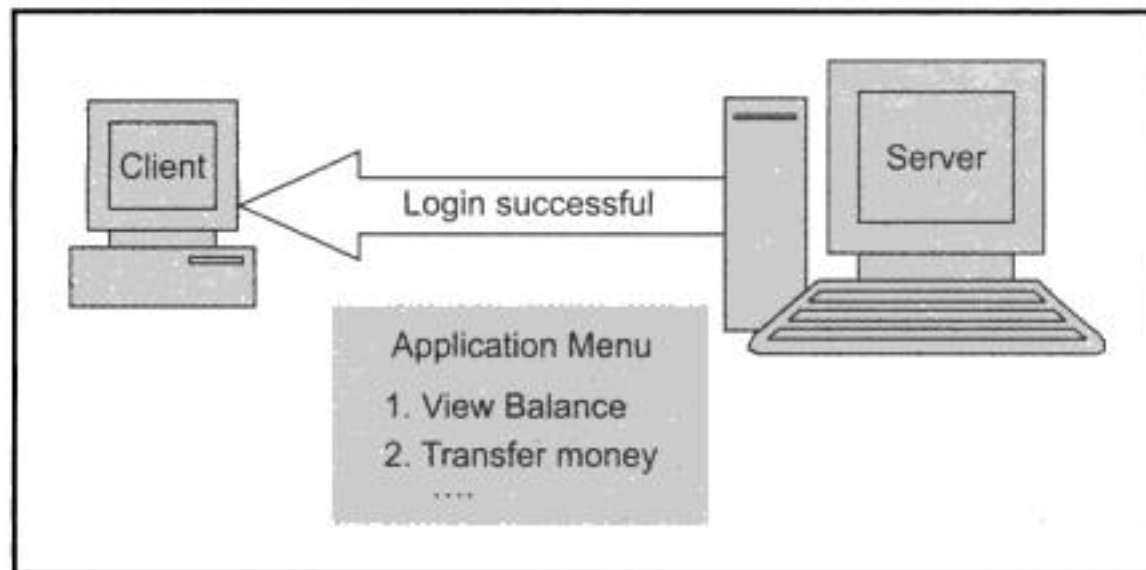


Fig. 7.34 Server compares the two random challenges

**Step 5: Server returns an appropriate message back to the user** Finally, the server sends an appropriate message back to the user, depending on whether the previous operations yielded success or failure. This is shown in Fig. 7.35.



— Fig. 7.35 Server sends an appropriate message back to the user

### 7.5.3 Use of Smart Cards

The use of smart cards can actually be related to certificate-based authentication. This is because smart cards allow the generation of private-public key pairs *within* the card. They also support the storage of digital certificates within the card. The private key always remains inside the card in a secure, tamper-free fashion. The public key and the certificate can be exported outside. Also, the smart cards are capable of performing cryptographic functions such as encryption, decryption, message digest creation and signing within the card.

Thus, during certificate-based authentication, the signing of random challenge sent by the server can be performed within the card. That is, the random challenge can be fed as input to the smart card, which can accept it, encrypt it with the smart card holder's private key, thus producing digital signature within the card, which it outputs back to the application.

However, one thing must be noted about the usage of smart cards. They must be used judiciously to perform selective cryptographic operations. For instance, if we wish to sign a 1 MB document using a smart card, it would be quite cumbersome if we expect the smart card to first produce a message digest of the document and then sign it. This is because just moving the 1 MB data to the smart card through the half-duplex 9,600 bits per second (bps) smart card interface would take about 15 minutes! Clearly, the approach should be to first generate a message digest outside of the smart card (i.e. inside the computer), and feed it to the smart card just for encrypting it to produce the digital signature.

Since smart cards are portable, one can virtually walk around with one's private key and digital certificate. Traditionally, smart cards have problems associated with them. These issues and their emerging solutions are listed in Table 7.1.

Still, there is a lack of standardization and inter-operability between smart card vendors. This would change, as the industry matures. This would mean that the security of PKI solutions can really become very strong.

**Table 7.1** *Problems and their Solutions Related to Smart Card Technology*

<i>Problem/Issue</i>	<i>Emerging solution</i>
Smart card readers are not yet a part of a desktop computer, unlike a hard disk drive or a floppy disk drive	The new versions of computers and mobile devices are expected to come with smart card readers <i>out of the box</i> .
Non-availability of smart card reader driver software	Microsoft has made the PC/SC smart card framework an integral part of the Windows 2000 operating system. Most smart card reader manufacturers ship the PC/SC compliant reader drivers, making the process of adding a reader hardware to the computer a plug-and-play operation.
Non availability of smart card aware cryptographic services software	Smart-card aware software such as Microsoft Crypto API (MS-CAPI) comes free with Internet Explorer.
Cost of smart cards and card readers is high	This is reducing now. Smart cards are available for about \$5, and the card readers for about \$20.

## 7.6 Biometric Authentication

### 7.6.1 Introduction

**Biometric authentication** mechanisms are receiving a lot of public attention. A biometric device is perhaps the ultimate attempt in trying to prove who you are. A biometric device works on the basis of some human characteristics, such as fingerprint, voice or pattern of lines in the iris of your eye. The user database contains a sample of user's biometric characteristics. During authentication, the user is required to provide another sample of the user's biometric characteristics. This is matched with the one in the database and if the two samples are the same, then the user is considered to be a valid one.

The important idea in biometrics is that the sample produced during every authentication process can vary slightly. This is because the physical characteristics of the user may change for a number of reasons. For instance, suppose the finger print of the user is captured and used for authentication every time. The sample taken every authentication may not be the same, because the finger can be dirty, can have cuts, other marks or the finger's position on the reader can be different and so on. Therefore, an exact match of the sample need not be required. An approximate match can be acceptable.

This is also the reason why, during the user registration process, multiple samples of the user biometric data are created. They are combined and their average stored in the user database, so that the different possibilities of the user's samples during the actual authentication can roughly map to this average sample. Using this basic philosophy, any biometric authentication system defines two configurable parameters: the **False Accept Ration (FAR)** and the **False Reject Ration (FRR)**. The FAR is a measurement of the chance that a user who should be rejected is actually accepted by a system as *good enough*. FRR is a measurement of the chance that a user who should be accepted as valid is actually rejected by a system as *not good enough*. Thus, FAR and FRR are exactly opposite of each other.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

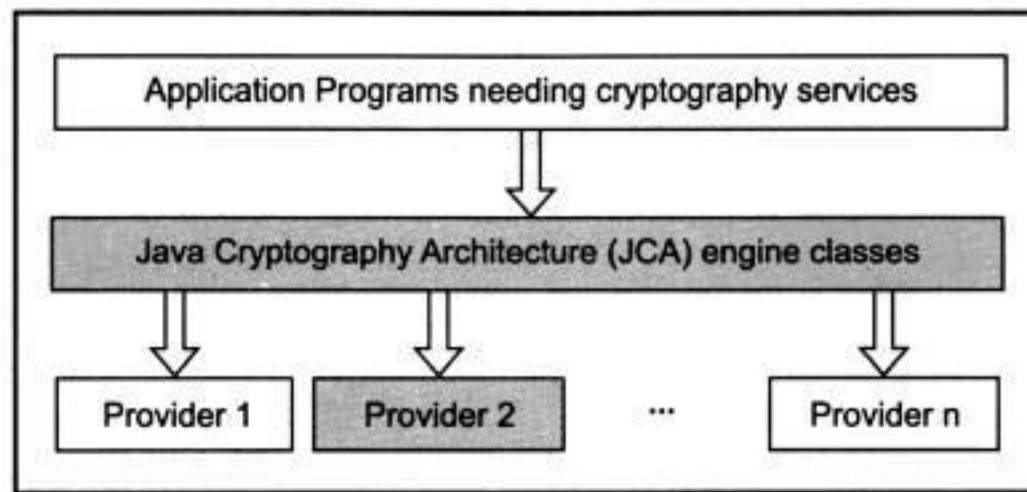




You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



— Fig. 8.4 JCA engine classes and providers

As we had discussed earlier, Java 2 came with a new utility called as **Keytool**. Keytool stores the public and private keys separately, and protects them with passwords. The database used by the keytool to store the keys is called as **keystore**. Usually, the keystore is a simple computer file with a *.keystore* extension, in the user's home directory. Let us list down a few important services provided by keytool:

- Creation of key pairs and self-signed certificates
- Export certificates
- Issue Certificate Signing Requests (CSR) to be sent to a Certification Authority (CA) for requesting a certificate
- Import other people's certificates for signature verification

For example, we can type *keytool -genkey* as a command to generate a key pair. Interestingly, the keystore database can be accessed even programmatically. For this, keystore is treated as a class and used in an application program.

**JCA Features** Having understood the basic concepts behind the JCA architecture, we will now discuss some of the features offered by JCA in terms of cryptographic capabilities.

As we might have realized by now, application developers are usually interested only in using the engine classes for performing the desired cryptographic operations. Each engine class has a public interface, i.e. a set of methods, which specify the operations that the engine class can perform. This is true for most Object Oriented (OO) systems these days, anyway. However, none of the engine classes has a public constructor. Instead, every engine class provides a *getInstance ( )* method, which accepts the name of the desired algorithm as an argument and returns an instance of the appropriate class.

Let us now look at an example of creating a message digest using the SHA-1 algorithm, written using JCA. The code contains detailed comments for those not familiar with the Java syntax, as shown in Fig. 8.5.

Let us summarize the steps involved in creating the message digest, as shown in the figure. The call to the *getInstance ( )* method finds and loads a message digest object that implements the SHA-1 message digest algorithm. After we create our input string to be digested, we pass that data to the *update ( )* method of the message digest object and then write it to the output file. Finally, we call the *digest ( )* method to create the message digest and add it to the same file.

Similar functionalities exist for digital signature and other cryptographic functionalities in JCA. In every case, the separation between the engine classes and the provider classes is carefully maintained and managed.

Even in the days of restrictions, things were not so simple. Companies had found loopholes to create their own *clones* of JCE as third party implementations. Of course, for the sake of completeness, it must be pointed out that JCE was not the only cryptography software that came under the restrictions imposed by the US government. Several other cryptographic software applications as well as the algorithms themselves were restricted. Moreover, many algorithms are patented as well. This means that the users of the algorithm must pay a licensing fee to the patent-holders. For instance, RSA Data Security Inc holds patents in the US on many algorithms based on RSA encryption and digital signature. Similarly, Ascom System AG (Switzerland) holds the patent for the IDEA algorithm. Therefore, if the application developers reside in a country where these patent rules apply, the application developers as well as the end users must pay a licensing fee to the patent holders, depending on the clauses in the patent document.

Coming back to the original discussion of the historical restrictions on JCE, Java application developers who wanted to use JCE had to observe the following points.

- They had to procure JCE separately from the JDK. The official JCE developed by Sun Microsystems could be obtained only by the citizens of the US and Canada. People residing in other countries could procure third-party implementations of JCE.
- Electronic documentation of JCE was also supposed to follow the same guidelines as above. (However, in practice, this was never the case. This clause was violated to a great extent).
- The JCE APIs and any applications developed using JCE could not be used outside the US or Canada. An interesting situation was: what if the developers hosted the application within the US or Canada and allowed applets to be downloaded on the browser clients outside of the US or Canada (which, in turn, used cryptography)? This could be highly possible in this age of Internet. Therefore, this was also restricted.

**JCE Architecture** The architecture of JCE follows the same pattern as that of JCA. It is also based on the concept of the engine classes and the provider classes. There is only one difference. JCE also comes with one implementation of the engine classes. This implementation is the default implementation, provided by Sun Microsystems. Since the architecture of JCE is very similar to that of JCA, we will not discuss it any further.

We will conclude our discussion with an example demonstrating an encryption process using JCE. This is shown in Fig. 8.6. As before, the example contains a lot of comments to help us understand what is going on inside the code.

The comments inside the code should have explained what is going on at each stage during the encryption process. We will not repeat those steps again.

#### 8.2.4 Conclusions

Both JCA and JCE are strong cryptographic architectures. They have been carefully planned and designed, so as to allow for future expansions as well as vendor-independence. However, the biggest problem with the use of Java cryptography was related to the licensing issues. Because of the US export laws, JCE did not come as a part of the core JDK, and it could also be procured outside the US and Canada easily. This was also why JCE was not a part of the Web browser software.

Now that the restrictions have been lifted, application developers can use JCE freely. The biggest advantage of using JCE is that it is free.

```

public class EncryptionDemo {
    public static void main (String args [ ] ) {
        try {
            // The KeyGenerator class provided by JCE can be used to generate
            // symmetric (secret) keys. We specify which algorithm we will use with
            // this symmetric key for actual encryption. Here, we specify it as DES.
            KeyGenerator kg = KeyGenerator.getInstance ("DES");

            // The Cipher class is used to instantiate an object of the specified
            // encryption algorithm class. We can also specify the mode and padding
            // scheme to be used during the encryption process. In this case, we indicate
            // that we want to use the DES encryption algorithm in the Cipher Block
            // Chaining (CBC) mode with padding as specified in PKCS#5 standard.
            Cipher c = Cipher.getInstance ("DES/CBC/PKCS5Padding");

            // The generateKey function generates a symmetric key, using the
            // parameters discussed above. The key is stored in a variable called as key.
            Key key = kg.generateKey ( );

            // JCE demands that once the key is generated, we must execute an init ( )
            // method against the Cipher object created earlier. This method takes two
            // parameters. The first parameter specifies if we want to perform
            // encryption or decryption. The second parameter specifies which key to
            // use in that operation.
            c.init (Cipher.ENCRYPT_MODE, key);

            // Now we specify the plain text, which we want to encrypt. We also
            // transform it into a byte array.
            byte plaintext [ ] = "I am plain text. Please encrypt me.".getBytes ( );

            // Execute the doFinal ( ) method, which performs the actual encryption or
            // decryption (in this case, it is encryption). It accepts the plain text as the
            // input parameter, and returns cipher text. Also, this method is a part of the
            // Cipher object (note the prefix c.).
            byte ciphertext [ ] = c.doFinal (plaintext);
        } catch (Exception e) {
            e.printStackTrace ( );
        }
    }
}

```

▮ Fig. 8.6 Example of encryption in Java using JCE



## 8.3 Cryptographic Solutions Using Microsoft .NET Framework

### 8.3.1 Class Model

We take a look at the cryptography features provided by Microsoft in its .NET framework.

Like JCA and JCE, the .NET framework cryptographic object model was designed to allow the addition of new algorithms and implementations in an effortless manner. In the model, a category of algorithms such as *symmetric algorithms* is modeled as a single abstract base class. Individual algorithms are represented by the respective abstract algorithm classes. Finally, there is a concrete implementation class per abstract algorithm class. Figure 8.7 shows the idea.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

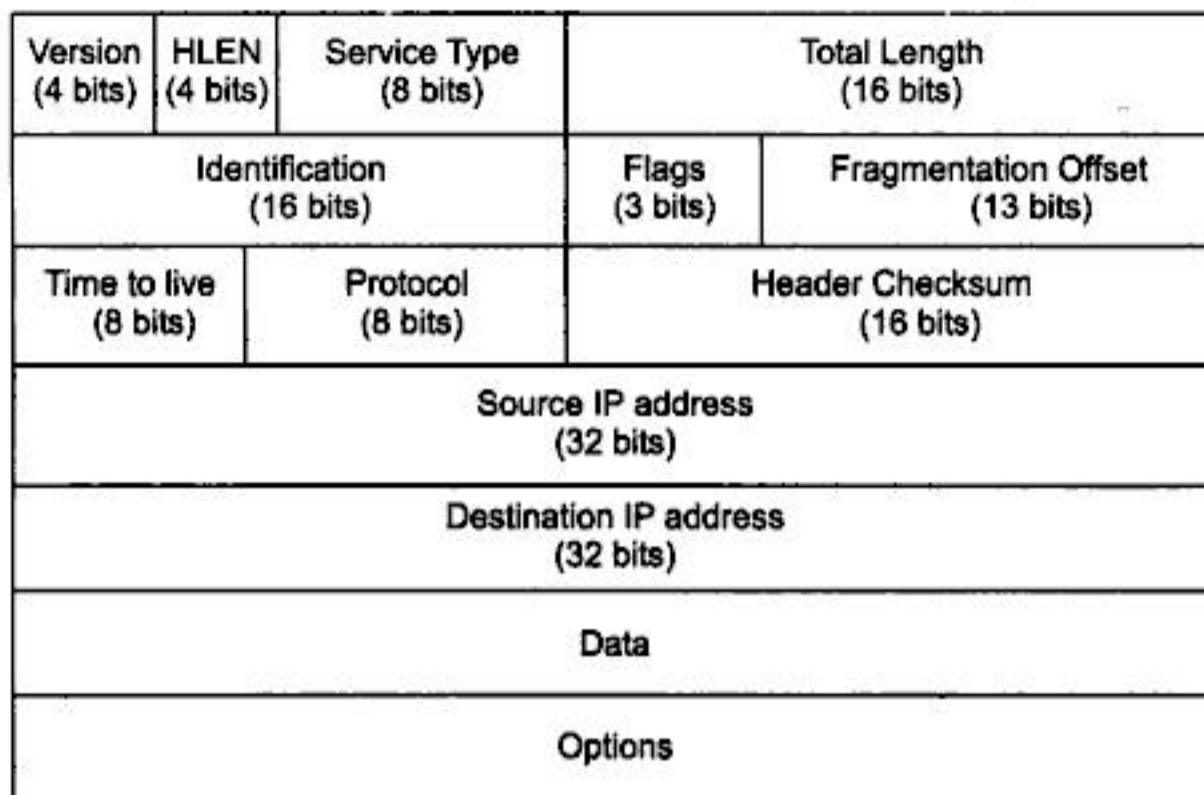
Let us briefly discuss the header fields inside a TCP segment.

- **Source port number:** This 2-byte number signifies the port number of the source computer, corresponding to the application that is sending this TCP segment.
- **Destination port number:** This 2-byte number signifies the port number of the destination computer, corresponding to the application that is expected to receive this TCP segment.
- **Sequence number:** This 4-byte field defines the number assigned to the first byte of the data portion contained in this TCP segment. TCP is a connection-oriented protocol. For ensuring a correct delivery, each byte to be transmitted from the source to the destination is numbered in an increasing sequence. The sequence number field tells the destination host, which byte in this sequence comprises the first byte of the TCP segment. During the TCP connection establishment phase, both the source as well as the destination generate different unique random numbers. For instance, if this random number is 3130 and the first TCP packet is carrying 2000 bytes of data, then the sequence number field for that packet would contain 3132 (bytes 3130 and 3131 are used in connection establishment). The second segment would then have a sequence number of 5132 ( $3132 + 2000$ ) and so on.
- **Acknowledgement number:** If the destination host receives a segment with sequence number  $X$  correctly, it sends  $X + 1$  as the acknowledgement number back to the source. Thus, this 4-byte number defines the sequence number that the source is expecting from the destination as a receipt of the correct delivery.
- **Header length:** This 4-bit field specifies the number of four-byte words in the TCP header. As we know, the header length can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 (because  $5 \times 4 = 20$ ) and 15 (because  $15 \times 4 = 60$ ).
- **Reserved:** This 6-byte field is reserved for future use and is currently unused.
- **Flag:** This 6-bit field defines six different control flags, each one of them occupying one bit. Out of the six flags, two are most important. The SYN flag indicates that the source wants to establish a connection with the destination. Therefore, this flag is used when a TCP connection is being established between two hosts. Similarly, the other flag of importance is the FIN flag. If the bit corresponding to this flag is set, then it means that the sender wants to terminate the current TCP connection.
- **Window size:** This field determines the size of the sliding window that the other party must maintain.
- **Checksum:** This 16-bit field contains the checksum for facilitating the error detection and correction.
- **Urgent pointer:** This field is used in situations where data in a TCP segment is more important or urgent than other data in the same TCP connection. However, a discussion of such situations is beyond the scope of the current text.

### 9.2.3 IP Datagram Format

The TCP header plus the original message is now passed to the IP layer. The IP layer treats this whole package of TCP header + original message as its *original message* and adds its own header to it. This results into the creation of an IP datagram. The format of an IP datagram is shown in Fig. 9.4.

An IP datagram is a variable-length datagram. A message can be broken down into multiple datagrams and a datagram in turn can be fragmented into different fragments, as we shall see. A



┆ Fig. 9.4 IP datagram

datagram can contain a maximum of 65,536 bytes. A datagram is made up of two main parts: the header and the data. The header consists of anywhere between 20 and 60 bytes and essentially contains information about routing and delivery. The data portion contains the actual data to be sent to the recipient. The header is like an envelope: it contains information *about* the data. The data is analogous to the letter *inside* the envelope. Let us examine the fields of a datagram in brief.

- **Version** – This field currently contains a value 4, which indicates **IP version 4 (IPv4)**. In future, this field would contain 6 when **IP version 6 (IPv6)** becomes the standard.
- **Header Length (HLEN)** – Indicates the size of the header in a multiple of four-byte words. When the header size is 20 bytes as shown in the figure, the value of this field is 5 (because  $5 \times 4 = 20$ ) and when the option field is at the maximum size, the value of HLEN is 15 (because  $15 \times 4 = 60$ ).
- **Service type** – This field is used to define service parameters such as the priority of the datagram and the level of reliability desired.
- **Total length** – This field contains the total length of the IP datagram. Because it is two bytes long, an IP datagram cannot be more than 65,536 bytes ( $2^{16} = 65,536$ ).
- **Identification** – This field is used in the situations when a datagram is fragmented. As a datagram passes through different networks, it might be fragmented into smaller sub-datagrams to match the physical datagram size of the underlying network. In these situations, the sub-datagrams are sequenced using the identification field, so that the original datagram can be reconstructed from them.
- **Flags** – This field corresponds to the earlier field (identification). It indicates whether a datagram can be fragmented in the first place – and if it can be fragmented, whether it is the first or the last fragment or it can be a middle fragment, etc.
- **Fragmentation offset** – If a datagram is fragmented, this field is useful. It is a pointer that indicates the offset of the data in the original datagram before fragmentation. This is useful when reconstructing a datagram from its fragments.

- **Time to live** – We know that a datagram travels through one or more routers before reaching its final destination. In the case of network problems, some of the routes to the final destination may not be available because of many reasons such as hardware failure, link failure or congestion. In that case, the datagram may be sent through a different route. This can continue for a long time if the network problems are not resolved quickly. Soon, there could be many datagrams traveling in different directions through lengthy paths, trying to reach their destinations. This can create congestion and the routers may become too busy, thus bringing at least parts of the Internet to a virtual halt. In some cases, the datagrams can continue to travel in a loop in between, without reaching the final destination and in fact, coming back to the original sender. To avoid this, the datagram sender initializes this field (that is, *Time to live*) to some number. As the datagram travels through routers, this field is decremented each time. If the value in this field becomes zero or negative, it is immediately discarded. No attempt is made to forward it to the next hop. This avoids a datagram traveling for an infinite amount of time through various routers and therefore, helps avoid network congestion. After all the other datagrams have reached the destination, the TCP protocol operating at the destination will find out this missing datagram and will have to request for its retransmission. Thus, IP is not responsible for the error-free, timely and in-sequence delivery of the entire message – it is done by TCP.
- **Protocol** – This field identifies the transport protocol running on top of IP. After the datagram is constructed from its fragments, it has to be passed on to the upper layer software piece. This could be TCP or UDP. This field specifies which piece of software at the destination node the datagram should be passed on to.
- **Source address** – This field contains the 32-bit IP address of the sender.
- **Destination address** – This field contains the 32-bit IP address of the final destination.
- **Options** – This field contains optional information such as routing details, timing, management and alignment. For instance, it can store the information about the exact route that the datagram has taken. When it passes through a router, the router puts in its id and optionally, also the time when it passed through that router, in one of the slots in this field. This helps tracing and fault detection of datagrams. However, most of the time, the space in this field is not sufficient for all these details, therefore, it is not used very often.

This brief introduction to TCP/IP would suffice for the scope of the current text.



## 9.3 Firewalls

### 9.3.1 Introduction

The dramatic rise and progress of the Internet has opened possibilities that no one would have thought of. We can connect any computer in the world to any other computer, no matter how far the two are located from each other. This is undoubtedly a great advantage for individuals and corporate as well. However, this can be a nightmare for network support staff, which is left with a very difficult job of trying to protect the corporate networks from a variety of attacks. At a broad level, there are two kinds of attacks:

- Most corporations have large amounts of valuable and confidential data in their networks. Leaking of this critical information to competitors can be a great setback.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

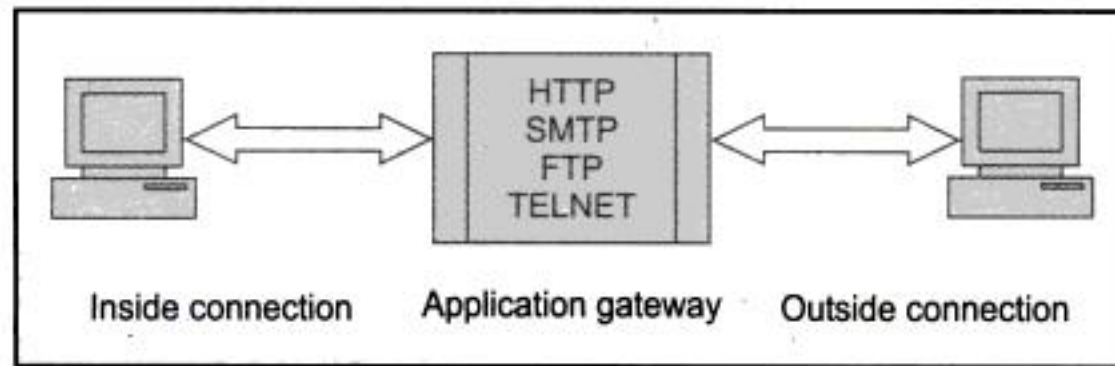


You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



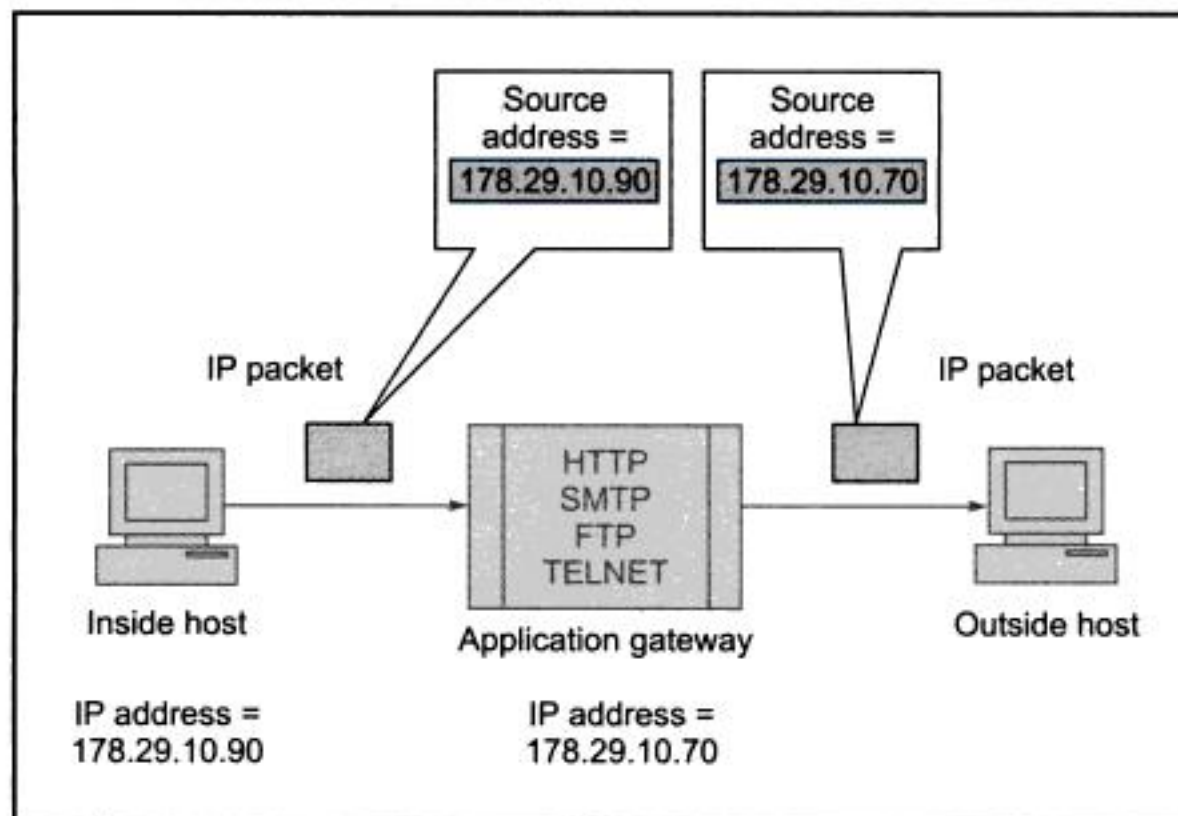
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





└ Fig. 9.13 Application gateway

4. The application gateway now accesses the remote host on behalf of the user and passes the packets of the user to the remote host. Note that there is a variation of the application gateway, called as **circuit gateway**, which performs some additional functions as compared to those performed by an application gateway. A circuit gateway, in fact, creates a new connection between itself and the remote host. The user is not aware of this and thinks that there is a direct connection between itself and the remote host. Also, the circuit gateway changes the source IP address in the packets from the end user's IP address to its own. This way, the IP addresses of the computers of the internal users are hidden from the outside world. This is shown in Fig. 9.14. Of course, both the connections are shown with a single arrow to stress on the concept, in reality, both the connections are two-ways.



└ Fig. 9.14 Circuit gateway operation

The SOCKS server is an example of the real-life implementation of a circuit gateway. It is a client-server application. The SOCKS client runs on the internal hosts and the SOCKS server runs on the firewall.

5. From here onwards, the application gateway acts like a proxy of the actual end user and delivers packets from the user to the remote host and vice versa.

Application gateways are generally more secure than packet filters, because rather than examining every packet against a number of rules, here we simply detect whether a user is allowed to work with

a TCP/IP application or not. The disadvantage of application gateways is the overhead in terms of connections. As we noticed, there are actually two sets of connections now: one between the end user and the application gateway and another between the application gateway and the remote host. The application gateway has to manage these two sets of connections and the traffic going between them. This means that the actual communicating internal host is under an illusion, as illustrated in Fig. 9.15.

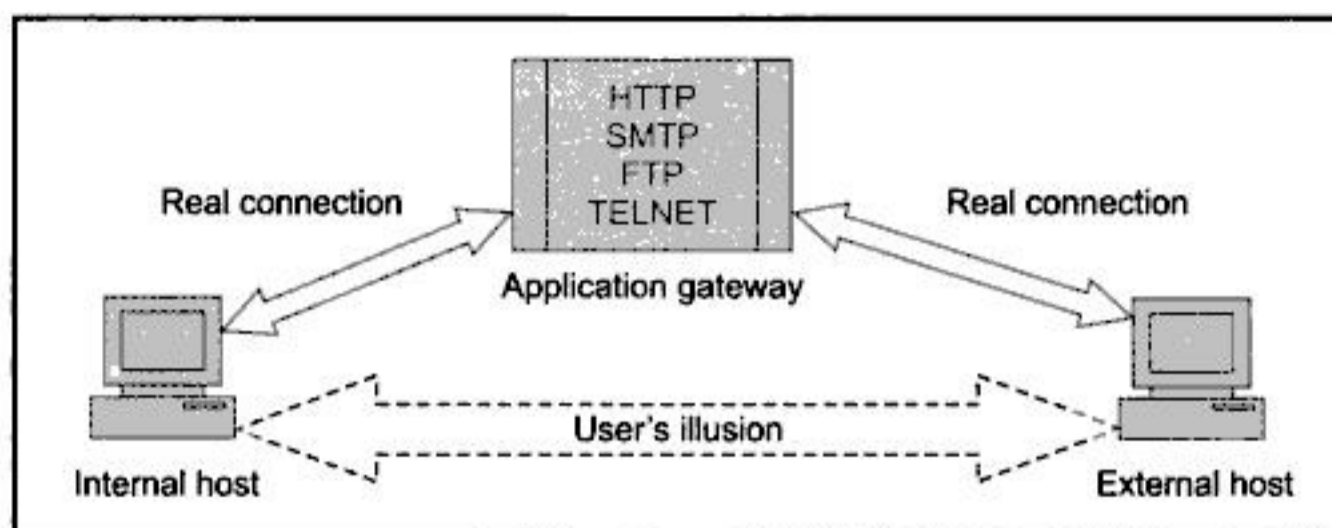


Fig. 9.15 Application gateway creates an illusion

An application gateway is also called as **bastion host**. Usually, a bastion host is a very key point in the security of a network. How it functions is explained in more detail in the next section.

**Network Address Translation (NAT)** One of the interesting jobs done by a firewall or proxy server is to perform **Network Address Translation (NAT)**. The number of people using the Internet from home, office or other places is increasing at a mind boggling rate. Earlier, users would access the Internet via an Internet Service Provider (ISP) for a short time and then disconnect. Thus, the ISP would have a set of IP addresses, from which it would dynamically allocate one IP address to every user for the duration the user was connected to the Internet. Once the user disconnected, the ISP would reallocate that same IP address to another user, who wanted to connect to the Internet now.

However, this situation changed dramatically as the number of people connecting to the Internet increased dramatically. Moreover, people started using the ADSL or cable connections to connect to the Internet, using the *broadband* technology. Worse yet, people wanted multiple IP addresses for themselves, since they started creating small personal networks. This led to a serious problem of shortage of IP addresses.

*NAT attempts to solve the problem of the shortage of IP addresses. NAT allows a user to have a large number of IP addresses internally, but only a single IP address externally. Only the external traffic needs the external address. The internal traffic can work with the internal addresses.*

For NAT to be possible, the Internet authorities have specified that certain IP addresses must be used as only internal IP addresses. Others should be used only as external IP addresses. Thus, just by looking at an IP address, we can determine whether it is an internal or external IP address. Also, routers and hosts have no confusion, because of this classification. The internal (or private) IP addresses are listed in Fig. 9.16.

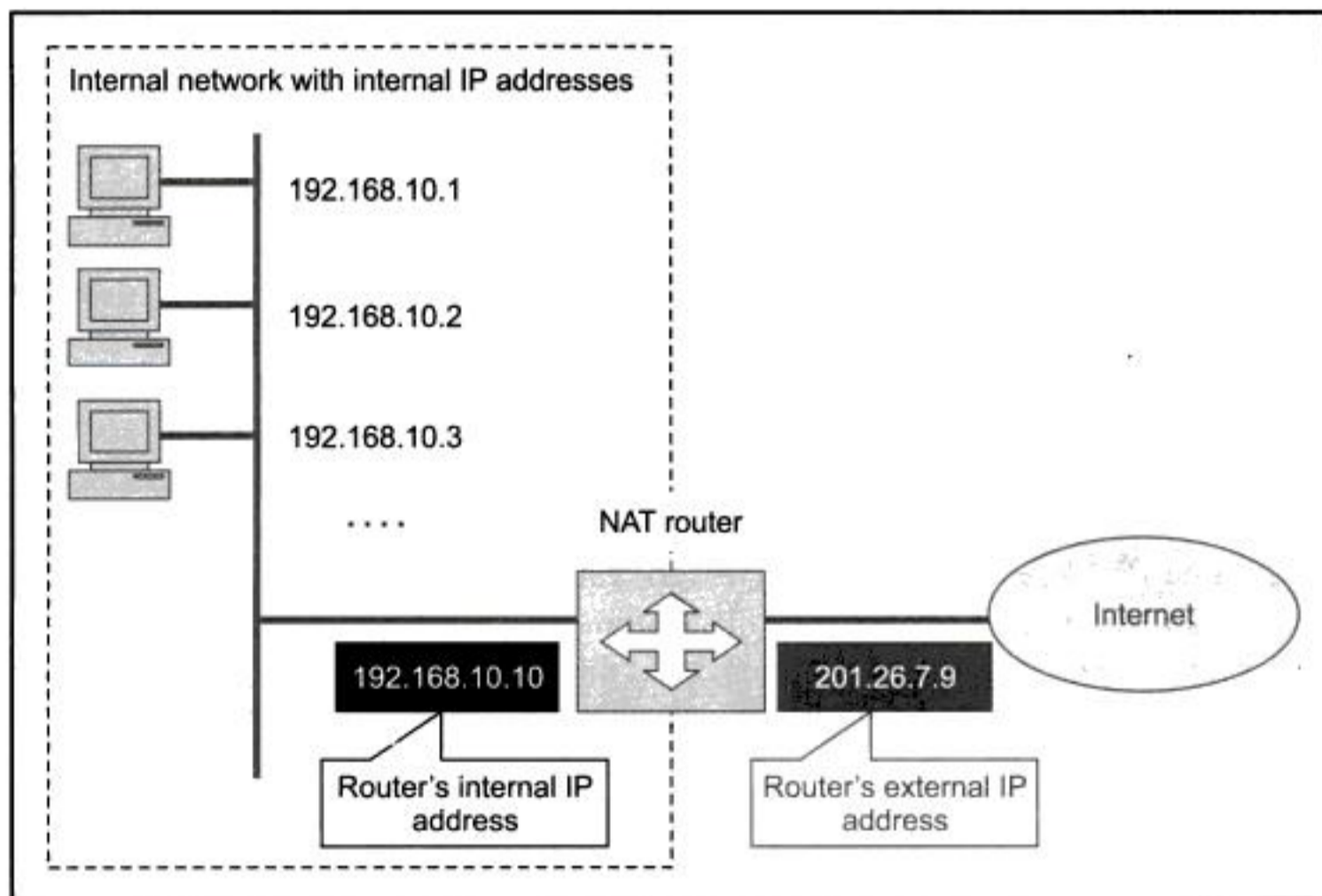
Any individual or organization can use any address within this range as an internal IP address, without needing to seek permission from anyone. Any address within this range is unique within that

organization's network, but is obviously not unique outside of the organization's network. That does not matter, since the address is meant to be used in the context of, i.e. inside an organization's network anyway. Therefore, if a router receives a packet with destination address equal to an address that falls in one of the above the ranges, it does not forward it outside, since it knows that the address is internal.

Range of IP addresses	Total count
10.0.0.0 to 10.255.255.255	$2^{24}$
172.16.0.0 to 172.31.255.255	$2^{20}$
192.168.0.0 to 192.168.255.255	$2^{16}$

┆ Fig. 9.16 Internal or private IP addresses

When implemented in real-life, a NAT configuration looks similar to what is shown in Fig. 9.17. As we can see, the router has two addresses: one external IP address and the other is an internal IP address. The external world (i.e. the rest of the Internet) knows the router based on the router's external address of 201.26.7.9, whereas the internal hosts refer to the router based on the router's internal address of 192.168.100.10. Also note how the internal hosts have internal IP addresses (192.168.x.x).



┆ Fig. 9.17 NAT implementation example

This clearly means that the external world always sees only one IP address: the NAT router's external IP address. Thus:

- For all *incoming* packets, regardless of which is actually the final destination in the internal network, the *destination address* field would always contain the NAT router's external address when the packet enters the network.
- For all *outgoing* packets, regardless of which is actually the original sender in the internal network, the *source address* field would always contain the NAT router's external address when the packet leaves the network.

As a result, the NAT router has to perform the job of address translation. For this purpose, the NAT router does the following:

- For all *incoming* packets, the NAT router replaces the *destination address* of the packet (which is set to the NAT router's external address) with the internal address of the final receiving host.
- For all *outgoing* packets, the NAT router replaces the *source address* of the packet (which is set to the internal address of the original sender host) with the external address of the NAT router.

This concept is shown in Fig. 9.18.

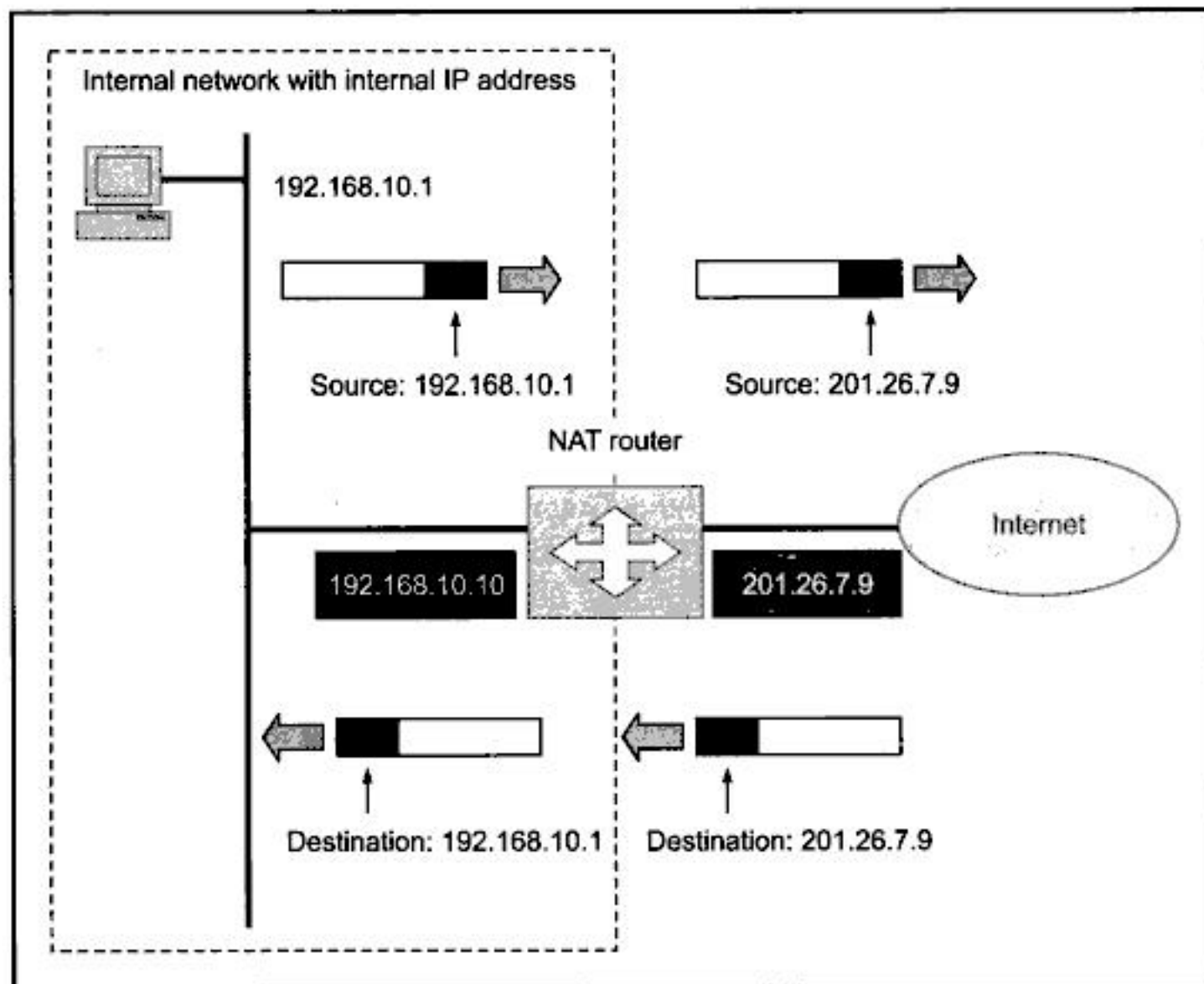


Fig. 9.18 NAT example

If we have studied this carefully, we would have realized that NAT for *outgoing packets* is pretty much straightforward. The NAT router simply has to replace the *source address* in the packet, which is the internal host's address with the external address of the NAT router. However, when it comes to *incoming packets*, how does the NAT router know what should be the actual internal host address? After all, if a network contains hundreds of hosts, the packet can be intended for any one of them!

For resolving this issue, the NAT router maintains a simple translation table, which maps the internal address of the host with the address of the external host to which the internal host is sending this packet. Thus, whenever an internal host sends a packet to an external host, the NAT makes an entry into the translation table. This entry contains the addresses of the internal host and that of the external host to which the packet is being sent over the Internet. Whenever a response comes back from any external host, the NAT router consults the translation table to see to which internal host the packet should be sent.

Let us consider an example to understand this.

- (a) Suppose an internal host (with address 192.168.10.1) wants to send a packet to an external host (with address 210.10.20.20). The internal host sends this packet on to the internal network, which reaches the NAT router. Currently, this packet contains *source address* = 192.168.10.1 and *destination address* = 210.10.20.20.
- (b) The NAT router adds an entry to the translation table, as follows:

Translation table	
Internal	External
192.168.10.1	210.10.20.20
...	...
...	...

- (c) The NAT router replaces the *source address* in the packet with its own address (i.e. 201.26.7.9) and sends the packet to the appropriate external host over the Internet, with the help of the usual routing mechanisms. Now, this packet contains *source address* = 201.26.7.9 and *destination address* = 210.10.20.20.
- (d) The external router processes the packet and sends a response back. Currently, this packet contains *source address* = 210.10.20.20 and *destination address* = 201.26.7.9.
- (e) The packet reaches the NAT router, as the destination address in the packet matches with that of the NAT router. The NAT router needs to find out whether this packet is meant for itself or for another internal host. Therefore, the NAT router consults its translation table to see if there is any entry for address 210.10.20.20 as the *External address*. In other words, the NAT router tries to find out if any host has sent a packet to and is expecting a response from an external host with address 210.10.20.20. It finds a match and comes to know that the internal host corresponding to this entry has an address of 192.168.10.1.
- (f) The NAT router replaces the *destination address* of the packet with that of the internal host for which it is destined, i.e. 192.168.10.1 and forwards the packet to this host.

This process is depicted in Fig. 9.19.

All this works fine, but we have another problem. With this scheme, only one internal host can communicate with any given external host at a given moment. Otherwise, the translation table will have multiple internal address entries for the same single external host. As a result, the NAT router will not be able to decide to which of these internal hosts a packet needs to be forwarded. In some cases, the NAT router has multiple external addresses. For example, if the NAT router has four external addresses, four internal hosts can access the same single external host now, each via a separate external NAT router address. However, there are two limitations in this approach:

- (i) There is still a limitation on the number of internal users that can access the same external host simultaneously.
- (ii) A single internal host cannot access two different applications on the same single external host (e.g. HTTP and FTP) at the same time. This happens because there is no way to distinguish between one application and another. For a single internal-external host combination, our translation table has a single entry.

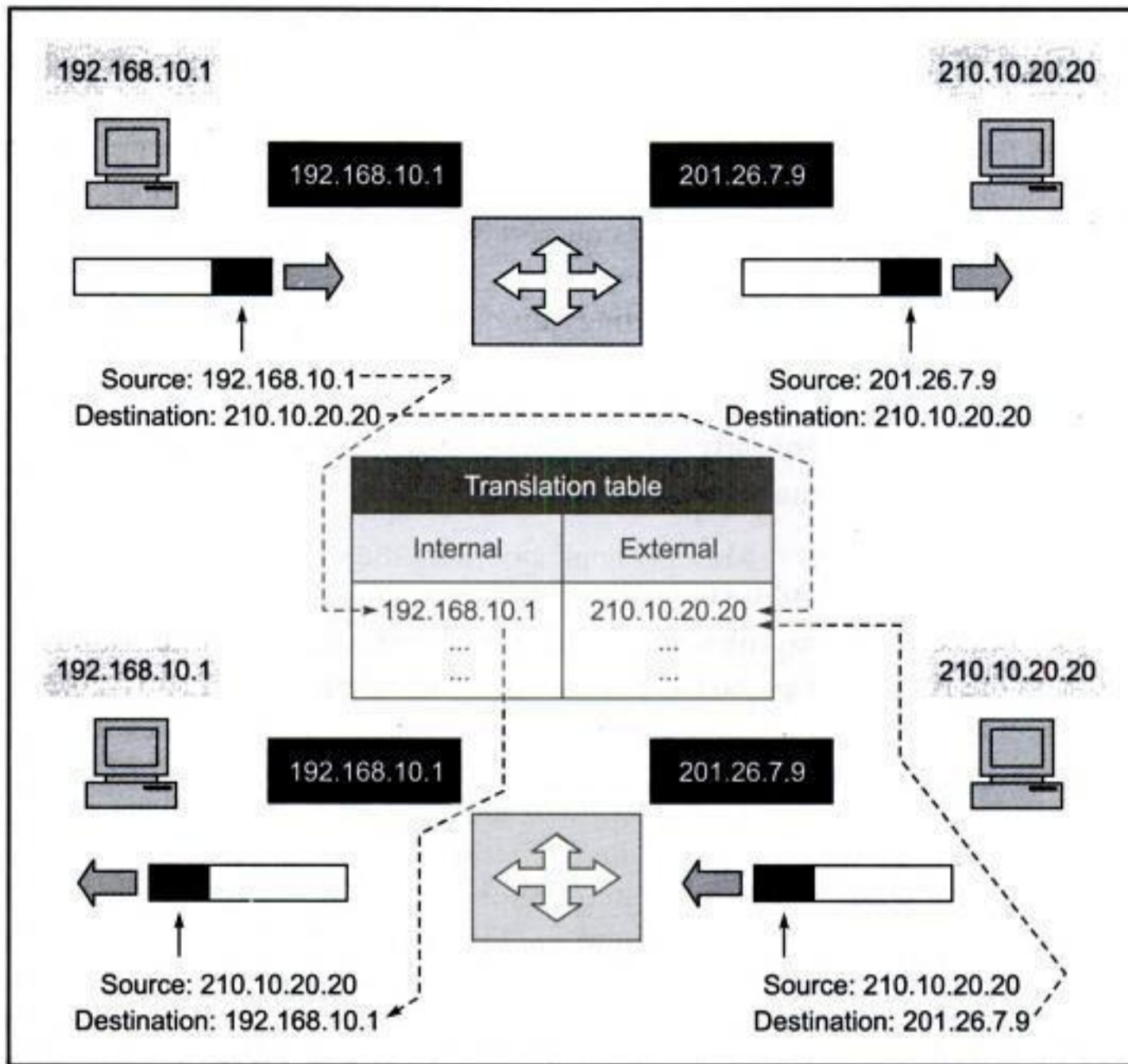


Fig. 9.19 Translation table in NAT

To resolve these issues, the translation table is modified to several new columns. The modified translation Table 9.1 gives further details.

Table 9.1 Modified NAT Translation Table

Internal address	Internal port	External address	External port	NAT port	Transport protocol
192.168.10.1	300	210.10.20.20	80	14000	TCP
192.168.10.1	301	210.10.20.20	21	14001	TCP
192.168.10.2	26601	210.10.20.20	80	14002	TCP
192.168.10.3	1275	207.21.1.5	80	14003	TCP

Let us now understand how these additional columns resolve our problems.

- The *Internal port* column signifies the port number used by the application program on the internal host. As per TCP/IP specifications, this port number is randomly chosen. However, this port

number is important because when a response comes back from the other side corresponding to the user's request, the user's computer needs to know to which application program the response needs to be handed over. This is determined by this port number.

- The *External port* column signifies the port number used by the application program on the server side. This port number is always fixed for a given application and is known as **well-known port**. For example, an HTTP server always runs on port 80, an FTP server always runs on port 21, etc. This is why we see these numbers in this column.
- The *NAT port* is a sequentially incrementing number, generated by the NAT router. This column has absolutely nothing to do with the source or destination port numbers. Instead, it is merely used like a primary key column in the translation table when a response comes back from the external host. This is explained subsequently.
- The *Transport protocol* column is not relevant here.

Let us consider both situations: (a) Multiple applications on the same internal host wanting to access the same external host and (b) Multiple internal hosts wanting to access the same external host.

Case (a) is shown in the first two rows of our modified translation table. Internal host 192.168.10.1 wants to work with the HTTP server and the FTP server of an external host 210.10.20.20. The internal host creates two port numbers 300 and 301 dynamically to open two connections with port numbers 80 and 21 respectively of the external host. When the packets move from the internal host to the NAT router, the NAT router as usual replaces the *source address* from that of the internal host to the NAT router's address. In addition, it replaces the *source port numbers* in the packets to 14000 and 14001, respectively and adds all these details to the translation table. The packets are then sent out to the external host 210.10.20.20, as usual. When the HTTP server on this external host sends back a response to the NAT router, the NAT router sees that the *destination port number* in this incoming packet is 14000. Therefore, it knows from the translation table that this packet needs to be sent to an internal host with address 192.168.10.1 on port 300. Similarly, when a response comes back from the FTP server of the same external host, the NAT router sees that the destination port in the packet is now 14001, looks up the translation table and forwards the packet to the same internal host 192.168.10.2, but on port 301.

Based on this discussion, it should be easy to imagine how case (b) is handled. The third row in our table has an entry for an internal host with address 192.168.10.2 and port number 26601 wanting to send a packet to the same external host 210.10.20.20, again on port 80. The NAT router changes the *source port number* in the packet as before, adds an entry to the translation table and sends the packet to the external host. When the external host responds, the NAT router uses the *destination port number* in this incoming packet to map it to the appropriate internal host and port number combination (i.e. 192.168.10.2 and 26601) using the translation table and sends it to that host.

Just for the sake of completeness, we have shown another internal host sending a packet to yet another external host.

### 9.3.3 Firewall Configurations

In practical implementations, a firewall is usually a combination of packet filters and application (or circuit) gateways. Based on this, there are three possible configurations of firewalls, as shown in Fig. 9.20.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



### 9.3.5 Limitations of Firewall

We must note that although a firewall is an extremely useful security measure for an organization, it does not solve all the practical security problems. The main limitations of a firewall can be listed as follows.

1. **Insider's intrusion:** As we know, a firewall system is designed to thwart outside attacks. Therefore, if an inside user attacks the internal network in some way; the firewall cannot prevent such an attack.
2. **Direct Internet traffic:** A firewall must be configured very carefully. It is effective only if it is the only entry-exit point of an organization's network. If, instead, the firewall is *one of the* entry-exit points, a user can bypass the firewall and exchange information with the Internet via the other entry-exit points. This can open up the possibilities of attacks on the internal network through those points. The firewall cannot, obviously, be expected to take care of such situations.
3. **Virus attacks:** A firewall cannot protect the internal network from virus threats. This is because a firewall cannot be expected to scan every incoming file or packet for possible virus contents. Therefore, a separate virus detection and removal mechanism is required for preventing virus attacks. Alternatively, some vendors bundle their firewall products with anti-virus software, to enable both the features *out of the box*.



## 9.4 IP Security

### 9.4.1 Introduction

The IP packets contain data in plain text form. That is, anyone watching the IP packets pass by can actually access them, read their contents and even change them. We have studied higher-level security mechanisms (such as SSL, SHTTP, PGP, PEM, S/MIME and SET) to prevent such kinds of attacks. Although these higher-level protocols enhance the protection mechanisms, there was a general feeling for a long time that why not secure IP packets themselves? If we can achieve this, then we need not rely only on the higher-level security mechanisms. The higher-level security mechanisms can then serve as additional security measures. Thus, we will have two levels of security in this scheme:

- First offer security at the IP packet level itself.
- Continue implementing higher-level security mechanisms, depending on the requirements.

This is shown in Fig. 9.25.

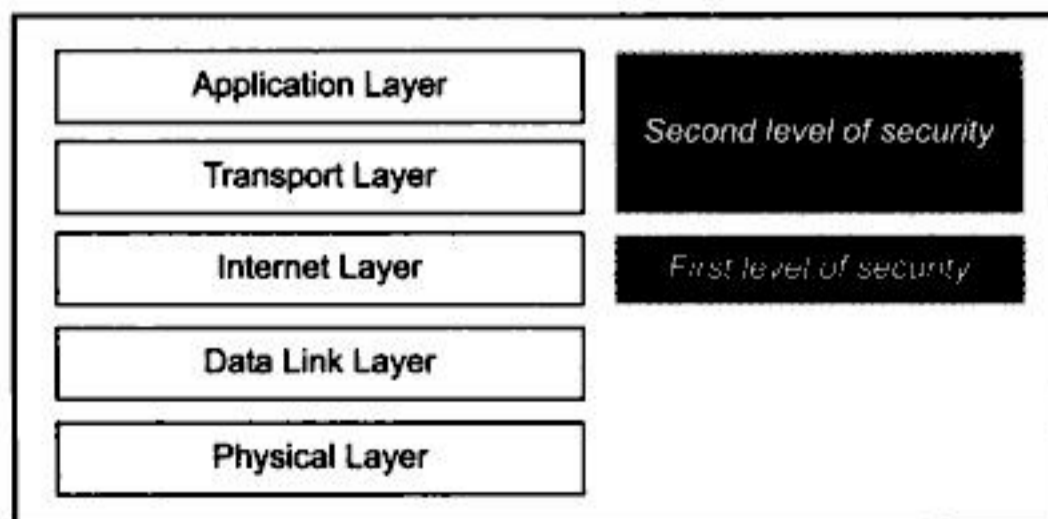


Fig. 9.25 Security at the Internet layer as well as the above layers

We have already discussed the higher-level security protocols. Our focus of discussion in this chapter is the first level of security (at the Internet layer).

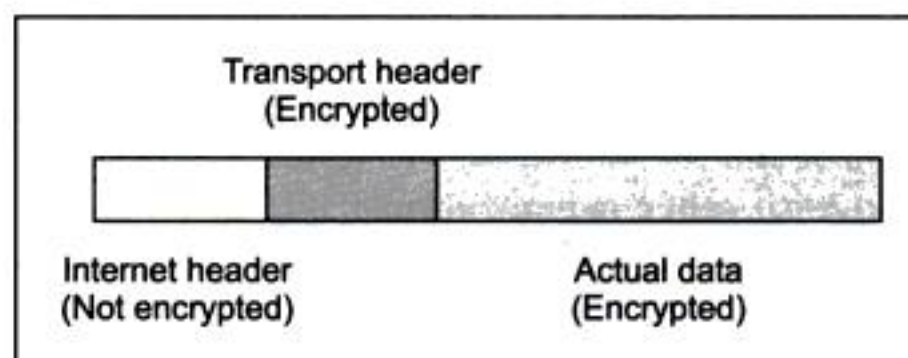
In 1994, the Internet Architecture Board (IAB) prepared a report, called as *Security in the Internet Architecture* (RFC 1636). This report stated that the Internet was a very open network, which was unprotected from hostile attacks. Therefore, said the report, the Internet needs better security measures, in terms of authentication, integrity and confidentiality. Just in 1997, about 150,000 Web sites were attacked in various ways, proving that the Internet was quite an unsafe place at times. Consequently, the IAB decided that authentication, integrity and encryption must be a part of the next version of the IP protocol, called as *IP version 6 (IPv6)* or *IP new generation (IPng)*. However, since the new version of IP was to take some years to be released and implemented, the designers devised ways to incorporate these security measures in the current version of IP, called as *IP version 4 (IPv4)* as well.

The outcome of the study and IAB's report is the protocol for providing security at the IP level, called as **IP Security (IPSec)**. In 1995, the Internet Engineering Task Force (IETF) published five security-based standards related to IPSec, as shown in Table 9.2.

**Table 9.2** RFC Documents Related to IPSec

RFC Number	Description
1825	An overview of the security architecture
1826	Description of a packet authentication extension to IP
1827	Description of a packet encryption extension to IP
1828	A specific authentication mechanism
1829	A specific encryption mechanism

IPv4 *may* support these features, but IPv6 *must* support them. The overall idea of IPSec is to encrypt and seal the transport and application layer data during transmission. It also offers integrity protection for the Internet layer. However, the Internet header itself is not encrypted, because of which the intermediate routers can deliver encrypted IPSec messages to the intended recipient. The logical format of a message after IPSec processing is shown in Fig. 9.26.



**Fig. 9.26** Result of IPSec processing

Thus, the sender and the receiver look at IPSec as shown in Fig. 9.27 as another layer in the TCP/IP protocol stack. This layer sits in-between the transport and the Internet layers of the conventional TCP/IP protocol stack.

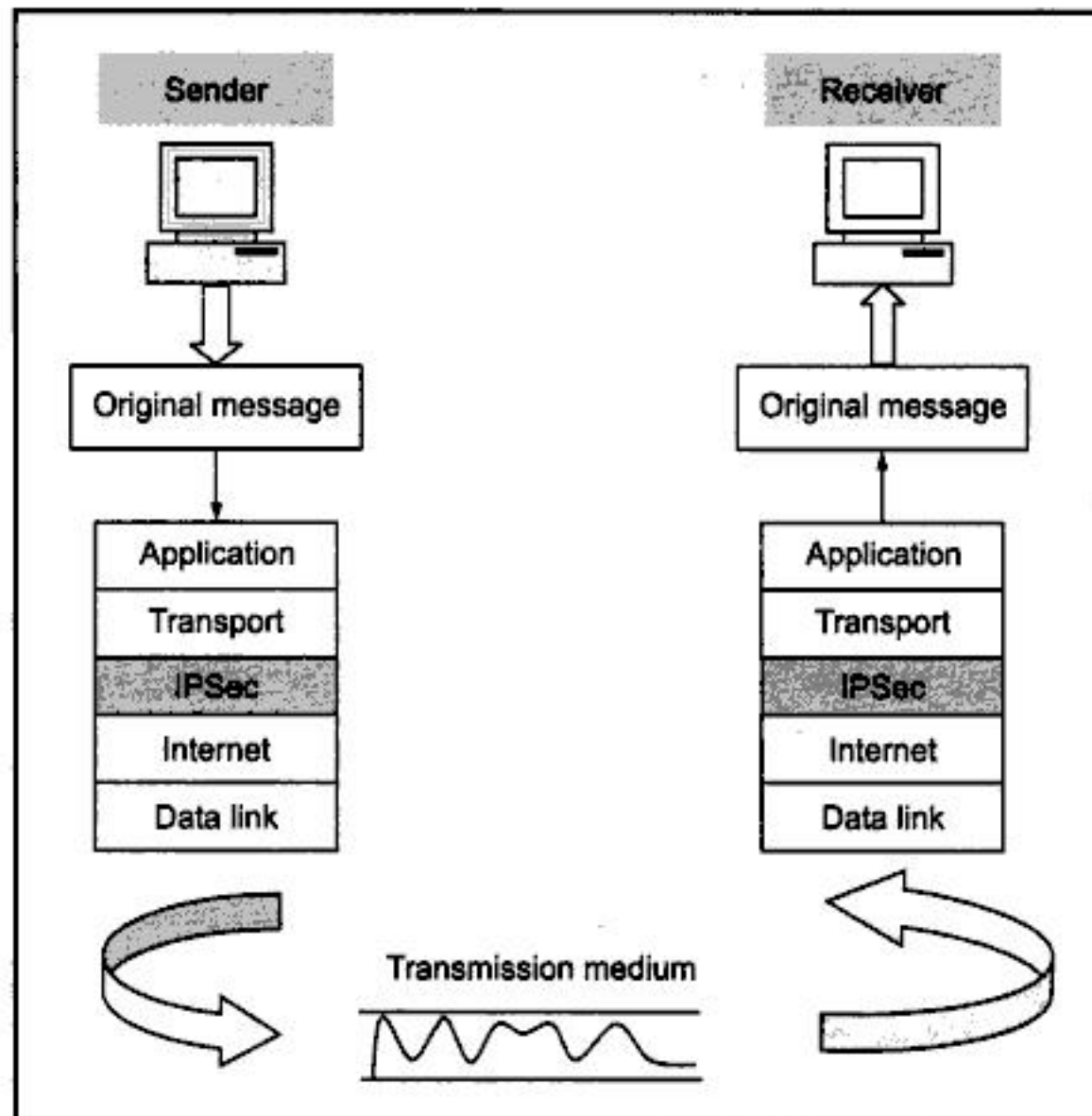


Fig. 9.27 Conceptual IPsec positioning in the TCP/IP protocol stack

## 9.4.2 IPsec Overview

**Applications and advantages** Let us first list the applications of IPsec.

- **Secure remote Internet access:** Using IPsec, we can make a local call to our Internet Service Provider (ISP) so as to connect to our organization's network in a secure fashion from our home or hotel. From there, we can access the corporate network facilities or access remote desktops/servers.
- **Secure branch office connectivity:** Rather than subscribing to an expensive leased line for connecting its branches across cities/countries, an organization can set up an IPsec-enabled network to securely connect all its branches over the Internet.
- **Set up communication with other organizations:** Just as IPsec allows connectivity between various branches of an organization, it can also be used to connect the networks of different organizations together in a secure and inexpensive fashion.

Following are the main advantages of IPsec.

- IPsec is transparent to the end users. There is no need for an user training, key issuance or revocation.
- When IPsec is configured to work with a firewall, it becomes the only entry-exit point for all traffic; making it extra secure.
- IPsec works at the network layer. Hence, no changes are needed to the upper layers (application and transport).

There are some more details that we should know. Both AH and ESP can be used in one of the two *modes*, as shown in Fig. 9.29.

We shall later study more about these modes. However, a quick overview would help.

In the **tunnel mode**, an encrypted *tunnel* is established between two hosts. Suppose X and Y are two hosts, wanting to communicate with each other using the IPSec tunnel mode. What happens here is that they identify their respective proxies, say P1 and P2 and a *logical encrypted tunnel* is established between P1 and P2. X sends its transmission to P1. The tunnel carries the transmission to P2. P2 forwards it to Y. This is shown in Fig. 9.30.

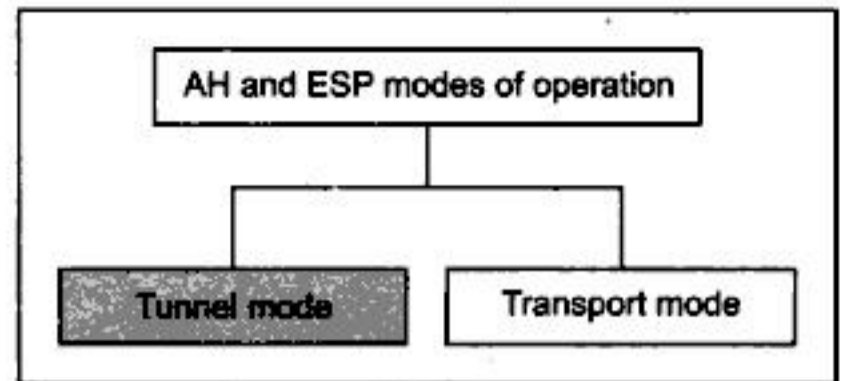


Fig. 9.29 AH and ESP modes of operation

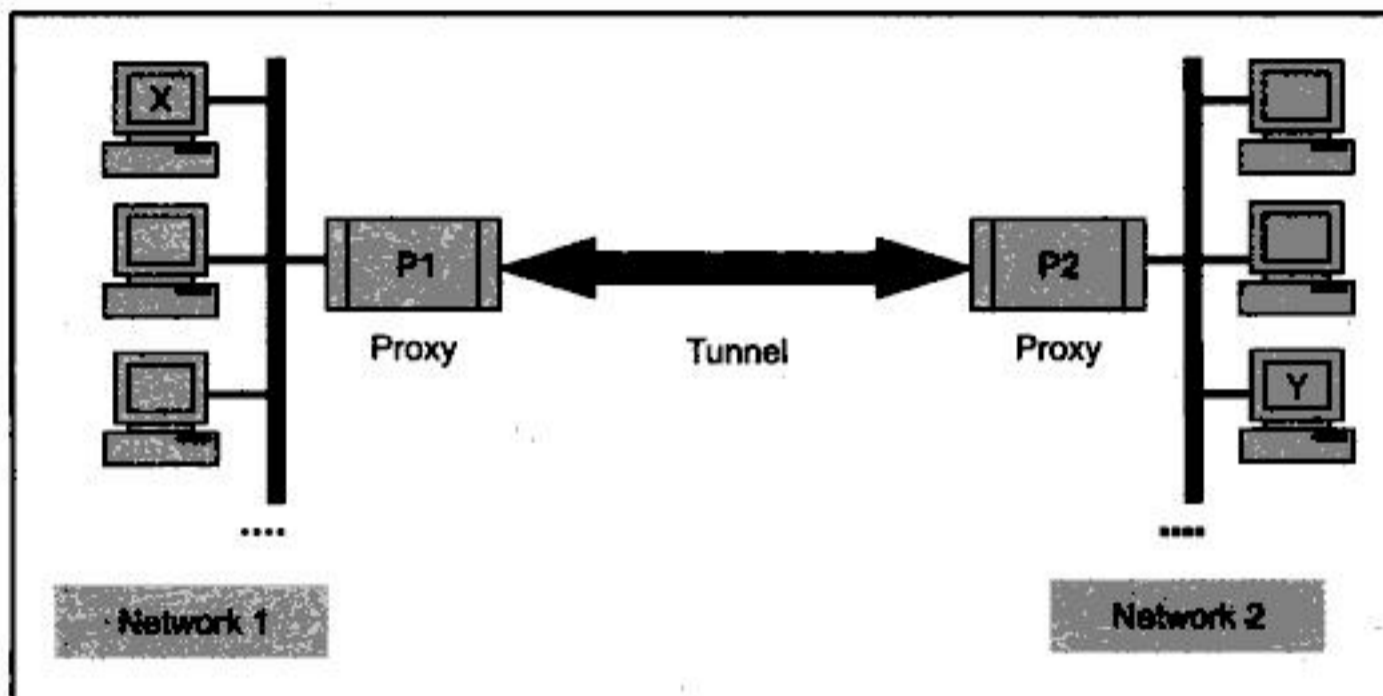


Fig. 9.30 Concept of tunnel mode

How do we implement this technically? As we shall see, we will have two sets of IP headers: internal and external. The internal IP header (which is encrypted) contains the source and destination addresses as X and Y, whereas the external IP header contains the source and destination addresses as P1 and P2. That way, X and Y are protected from potential attackers. This is shown in Fig. 9.31.

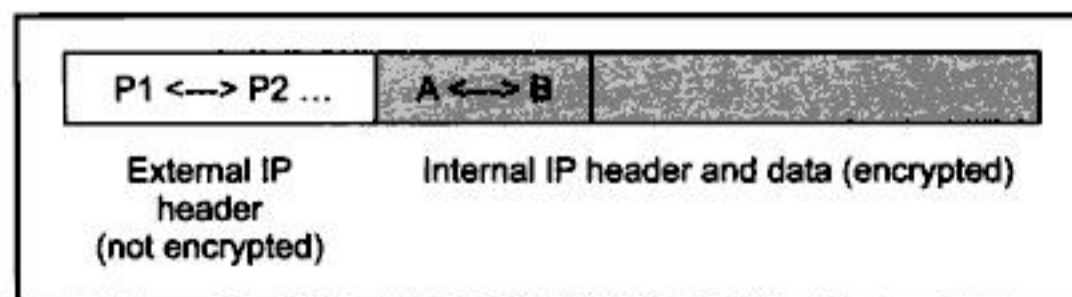


Fig. 9.31 Implementation of tunnel mode

- In the tunnel mode, IPSec protects the entire IP datagram. It takes an IP datagram (including the IP header), adds the IPSec header and trailer and encrypts the whole thing. It then adds new IP header to this encrypted datagram.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



### 9.4.5 IPsec Key Management

**Introduction** Apart from the two core protocols (AH and ESP), the third most significant aspect of IPsec is key management. Without a proper key management set up, IPsec cannot exist. This key management in IPsec consists of two aspects: key agreement and distribution. As we know, we require four keys if we want to make use of both AH and ESP: two keys for AH (one for message transmissions, one for message receiving) and two keys for ESP (one for message transmissions, one for message receiving).

The protocol used in IPsec for key management is called as **ISAKMP/Oakley**. The **Internet Security Association and Key Management Protocol (ISAKMP)** protocol a platform for key management. It defines the procedures and packet formats for negotiating, establishing, modifying and deleting SAs. ISAKMP messages can be transmitted via the TCP or UDP transport protocol. TCP and UDP port number 500 is reserved for ISAKMP.

The initial version of ISAKMP mandated the use of the Oakley protocol. Oakley is based on the Diffie-Hellman key exchange protocol, with a few variations. We will first take a look at Oakley and then examine ISAKMP.

**Oakley Key Determination Protocol** The Oakley protocol is a refined version of the Diffie-Hellman key exchange protocol. We will not repeat the concepts of Diffie-Hellman, as we have already studied it in great detail. However, we will note here that Diffie-Hellman offers two desirable features:

- (a) Creation of secret keys as and when required
- (b) No requirement for any preexisting infrastructure

However, Diffie-Hellman also suffers from a few problems, as follows:

- No mechanism for authentication of the parties.
- Vulnerability to man-in-the-middle-attack.
- Involves a lot of mathematical processing. An attacker can take undue advantage of this by sending a number of hoax Diffie-Hellman requests to a host. The host can unnecessarily spend a large amount of time in trying to compute the keys, rather than doing any actual work. This is called as **congestion attack** or **clogging attack**.

The Oakley protocol is designed to retain the advantages of Diffie-Hellman and to remove its drawbacks. The features of Oakley are as follows.

1. It has features to defeat replay attacks.
2. It implements a mechanism called as cookies to defeat congestion attacks.
3. It enables the exchange of Diffie-Hellman public key values.
4. It provides authentication mechanisms to thwart man-in-the-middle attacks.

We have already discussed the Diffie-Hellman key exchange protocol in great detail. Here, we shall simply discuss the approaches taken by Oakley to tackle the issues with Diffie-Hellman.

- **Authentication** Oakley supports three authentication mechanisms: digital signatures (generation of a message digest and its encryption with the sender's private key), public key encryption (encrypting some information such as the sender's user id with the recipient's public key) and secret key encryption (a key derived by using some out-of-band mechanisms).

- **Dealing with congestion attacks** Oakley uses the concept of cookies to thwart congestion attacks. As we know, in this kind of attack, an attacker forges the source address of another legitimate user and sends a public Diffie-Hellman key to another legitimate user. The receiver performs modular exponentiation to calculate the secret key. A number of such calculations performed rapidly one after the other can cause congestion or clogging of the victim's computer. To tackle this, each side in Oakley must send a pseudo-random number, called as cookie, in the initial message, which the other side must acknowledge. This acknowledgement must be repeated in the first message of Diffie-Hellman key exchange. If an attacker forges the source address, she does not get the acknowledgement cookie from the victim and her attack fails. Note that at the most the attacker can force the victim to generate and send a cookie, but not to perform the actual Diffie-Hellman calculations.

The Oakley protocol provides for a number of message types. For simplicity, we shall consider only one of them, called as *aggressive key exchange*. It consists of three message exchanges between the two parties, say X and Y. Let us examine these three messages.

- **Message 1:** To begin with, X sends a cookie and the public Diffie-Hellman key of X for this exchange, along with some other information. X signs this block with its private key.
- **Message 2:** When Y receives *message 1*, it verifies the signature of X using the public key of X. When Y is satisfied that the message indeed came from X, it prepares an acknowledgement message for X, containing the cookie sent by X. Y also prepares its own cookie and Diffie-Hellman public key and along with some other information, it signs the whole package with its private key.
- **Message 3:** Upon receipt of *message 2*, X verifies it using the public key of Y. When X is satisfied about it, it sends a message back to Y to inform that it has received Y's public key.

**ISAKMP** The ISAKMP protocol defines procedures and formats for establishing, maintaining and deleting SA information. An ISAKMP message contains an ISAKMP header followed by one or more payloads. The entire block is encapsulated inside a transport segment (such as TCP or UDP segment). The header format for ISAKMP messages is shown in Fig. 9.44.

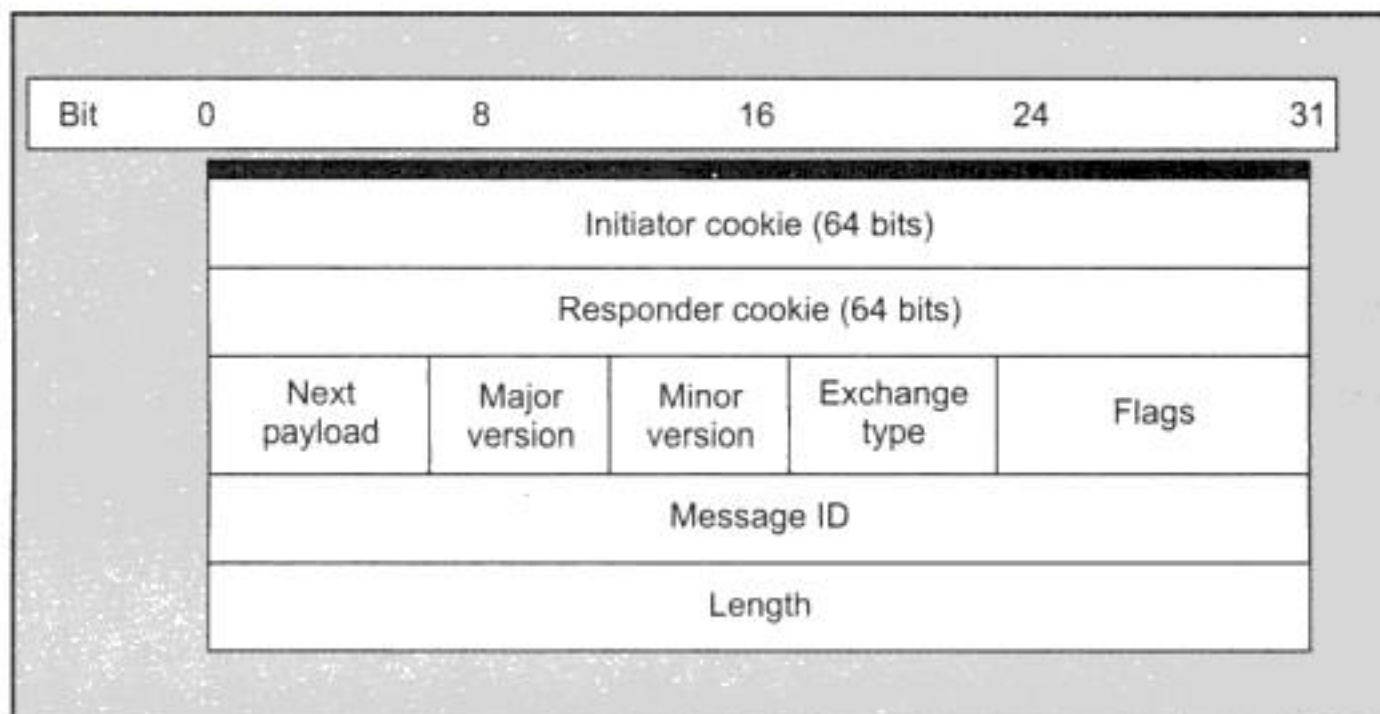


Fig. 9.44 ISAKMP header format

Let us assume that an organization wants to connect two of its branch networks to each other. The trouble is that these branches are located quite a distance apart. One branch is in Delhi and the other branch is in Mumbai. Two solutions out of all the available ones seem logical:

- Connect the two branches using a personal network, i.e. lay cables between the two offices yourself or obtain a leased line between the two branches.
- Connect the two branches with the help of a public network, such as the Internet.

The first solution gives far more control and offers a sense of security as compared to the second solution. However, it is also quite complicated. Laying cables between two cities is not easy and is usually not permitted either. The second solution seems easier to implement, as there is no special infrastructure set up required. However, it also seems to be vulnerable to possible attacks. How nice it would be, if we could combine the two solutions!

**Virtual Private Networks (VPN)** offers such a solution. A VPN is a mechanism of employing encryption, authentication and integrity protection so that we can use a public network (such as the Internet) as if it is a private network (such as a physical network created and controlled by you). VPN offers high amount of security and yet does not require any special cabling on behalf of the organization that wants to use it. Thus, a VPN combines the advantages of a public network (cheap and easily available) with those of a private network (secure and reliable).

A VPN can connect distant networks of an organization or it can be used to allow traveling users to remotely access a private network (e.g. the organization's intranet) securely over the Internet.

A VPN is thus a mechanism to simulate a private network over a public network, such as the Internet. The term *virtual* signifies that it depends on the use of virtual connections. These connections are temporary and do not have any physical presence. They are made up of packets.

### 9.5.2 VPN Architecture

The idea of a VPN is actually quite simple to understand. Suppose an organization has two networks, *Network 1* and *Network 2*, which are physically apart from each other and we want to connect them using the VPN approach. In such a case, we set up two firewalls, *Firewall 1* and *Firewall 2*. The encryption and decryption are performed by the firewalls. The architectural overview is shown in Fig. 9.45.

We have shown two networks, *Network 1* and *Network 2*. *Network 1* connects to the Internet via a firewall named *Firewall 1*. Similarly, *Network 2* connects to the Internet with its own firewall, *Firewall 2*. We shall not worry about the configuration of the firewall here and shall assume that the best possible configuration is selected by the organization. However, the key point here is that the two firewalls are *virtually* connected to each other via the Internet. We have shown this with the help of a *VPN tunnel* between the two firewalls.

With this configuration in mind, let us understand how the VPN protects the traffic passing between any two hosts on the two different networks. For this, let us assume that host X on *Network 1* wants to send a data packet to host Y on *Network 2*. This transmission would work as follows.

1. Host X creates the packet, inserts its own IP address as the source address and the IP address of host Y as the destination address. This is shown in Fig. 9.46. It sends the packet using the appropriate mechanism.

3. The packet reaches *Firewall 2* over the Internet, via one or more routers, as usual. *Firewall 2* discards the outer header and performs the appropriate decryption and other cryptographic functions as necessary. This yields the original packet, as was created by host X in Step 1. This is shown in Fig. 9.48. It then takes a look at the plain text contents of the packet and realizes that the packet is meant for host Y (because the destination address inside the packet specifies host Y). Therefore, it delivers the packet to host Y.

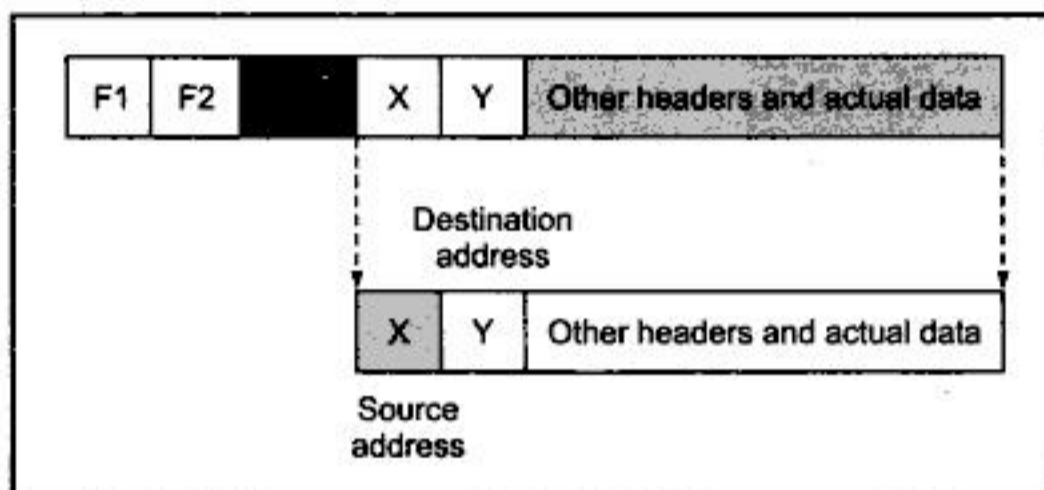


Fig. 9.48 Firewall 2 retrieves the original packet contents

There are three main VPN protocols. A detailed study of these protocols is beyond the scope of the current text. However, we shall briefly discuss them for the sake of completeness.

- The **Point to Point Tunneling Protocol (PPTP)** is used on Windows NT systems. It mainly supports the VPN connectivity between a single user and a LAN, rather than between two LANs.
- Developed by IETF, the **Layer 2 Tunneling Protocol (L2TP)** is an improvement over PPTP. L2TP is considered as the secure open standard for VPN connections. It works for both combinations: user-to-LAN and LAN-to-LAN. It can include the IPsec functionality as well.
- Finally, IPsec can be used in isolation. We have discussed IPsec in detail earlier.

## 9.6 Intrusion

### 9.6.1 Intruders

No matter how much secure a system is made, there would be attackers, who would constantly try to find their way. We call them **intruders**, because they try to intrude into the privacy of a network. Whether the network itself is private (e.g. a Local Area Network) or public (the Internet) does not matter. What matters is the intent of the attacker, of trying to intrude. It is generally said that the two most widely known threats to security are intruders and viruses. We shall concentrate on intruders here.

Intruders are said to be of three types, as explained below:

- **Masquerader:** A user who does not have the authority to use a computer, but penetrates into a system to access a legitimate user's account is called as a **masquerader**. It is generally an external user.
- **Misfeasor:** There are two possible cases for an internal user to be called as a **misfeasor**:
  - o A legitimate user, who does not have access to some applications, data or resources accesses them.

- o A legitimate user, who has access to some applications, data or resources misuses these privileges.
- **Clandestine user:** An internal or external user who tries to work using the privileges of a supervisor user to avoid auditing information being captured and recorded is called as a **clandestine user**.

How do intruders try to attack? A simple example may be considered, where the attackers try to obtain the passwords of legitimate users, so as to impersonate them. Some of the popularly known methods of password guessing are as follows:

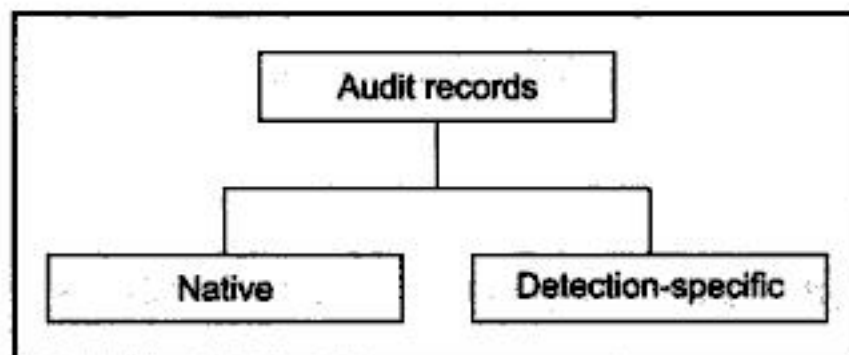
1. Try all possible short password combinations (2-3 characters).
2. Collect information about users, such as their full name, names of family members, their hobbies, etc.
3. Try default passwords that are provided by the supplier of a software product (e.g. Oracle comes with *scott* as the user name and *tiger* as the password).
4. Try words that people choose as passwords most often. Hacker bulletin boards maintain these lists. Also, try words from dictionary.
5. Try using phone numbers, dates of birth, social security numbers, bank account numbers, etc.
6. Tap the communication line between a user and the host network.
7. Use a Trojan Horse.
8. Try numbers on the vehicle license plates.

Regardless of how the intruder gets into a system, we need to first try and prevent it, if not, atleast detect it and take an appropriate action.

### 9.6.2 Audit Records

One of the most important tools in intrusion detection is the usage of **audit records**, also called as **audit logs**. Audit records are used to record information about the actions of users. Traces of illegitimate user actions can be found in these records, so as to detect intrusions so as to take appropriate actions.

Audit records can be classified into two categories: **Native audit records** and **Detection-specific audit records**. This is shown in Fig. 9.49.



┆ Fig. 9.49 Audit records classification

- **Native audit records:** All multi-user operating systems have accounting software built-in. This software records information about all user actions.
- **Detection-specific audit records:** This type of audit records facility collects information specific only to intrusion detection. This is more focused, but may duplicate information.

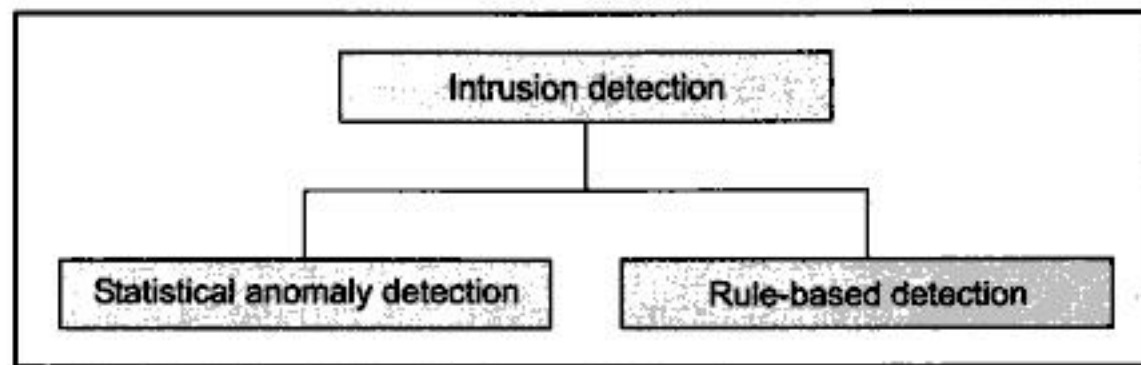


Fig. 9.50 Classification of intrusion detection

- o **Threshold detection:** In this type, thresholds are defined for all the users as a group and frequency of various events is measured against these thresholds.
- o **Profile-based detection:** In this type, profiles for individual users are created and they are matched against the collected statistics to see if any irregular patterns emerge.
- **Rule-based detection:** A set of rules is applied to see if a given behavior is suspicious enough to be classified as an attempt to intrude. This is also classified into two sub-types:
  - o **Anomaly detection:** Usage patterns are collected to analyze deviation from these usage patterns, with the help of certain rules.
  - o **Penetration identification:** This is an expert system that looks for illegitimate behavior.

#### 9.6.4 Distributed Intrusion Detection

Focus has started moving from intrusion detection on single systems to distributed systems, e.g. a LAN or a WAN. Following factors are important in this scheme of **distributed intrusion detection**:

- Different systems in the distributed system may record audit information in different formats. This needs to be uniformly processed.
- Typically one or a few nodes on the distributed system would be used to gather and analyze audit information. Hence, there should be provisions to securely send audit information from all other hosts to these hosts.

#### 9.6.5 Honeypots

Modern intrusion detection systems make use of a novel idea, called as **honeypots**. A honeypot is a trap that attracts potential attackers. A honeypot is designed so as to do the following:

- Divert the attention of a potential intruder from critical systems
- Collect information about the intruder's actions
- Provide encouragement to the intruder so as to stay on for some time, allowing the administrators to detect this and swiftly act on it

Honeypots are designed with two important goals in mind:

- (a) Make them look like real-life systems. Put as much of real-looking (but fabricated) information into them as possible.
- (b) Do not allow legitimate users to know about or access them.

Naturally, anyone trying to access a honeypot is a potential intruder. Honeypots are armed with sensors and loggers, which alarm the administrators of any user actions.



## SUMMARY

- ❑ Corporate networks can be attacked from outside or internal information can be leaked out.
- ❑ Encryption cannot prevent outside attackers from attacking a network.
- ❑ A firewall should be placed between a corporate network and the outside world.
- ❑ A firewall is a special type of router, which applies rules for allowing or stopping traffic.
- ❑ A firewall stands like a sentry on the main door between the internal network and the external Internet.
- ❑ A firewall can be application gateway or packet filter.
- ❑ A packet filter examines every packet, applies rules and decides whether to allow the traffic to proceed or not.
- ❑ Packet filters are quite useful in applying more specific/granular rules.
- ❑ Dynamic packet filter (also called as stateful packet filter) adapts itself to the changing conditions.
- ❑ An application gateway works at the application layer. It decides whether to allow traffic for a certain application (e.g. HTTP or FTP). It does not apply granular rules such as *Stop the packet if the source IP address is x.x.x.x*, the way packet filter works.
- ❑ A circuit gateway creates a new connection between itself and the remote host.
- ❑ Firewall architectures combine the various types of firewalls in some combination.
- ❑ Screened subnet firewall is the strongest firewall architecture.
- ❑ Network Address Translation (NAT) allows a few IP addresses to be shared across many networks in the world, thus saving on the IP address space.
- ❑ Without NAT, the available range of IP addresses would have exhausted long back.
- ❑ NAT classifies certain IP addresses as internal, which have no recognition outside that network.
- ❑ Internal addresses are unique inside a network but are duplicated across different networks.
- ❑ A NAT router performs the job of translating between an internal and an external address.
- ❑ The NAT router has to maintain a translation table and use some intelligent tricks to perform address translation.
- ❑ IPSec provides security between the transport and the Internet layers.
- ❑ IPSec provides authentication and confidentiality services.
- ❑ A Demilitarized Zone (DMZ) firewall protects both the servers of an organization that are to be exposed to the external world, as well as the internal corporate network, which needs to be secured from the external world.
- ❑ IPSec protocol is used to apply security at the network layer.
- ❑ IPSec does not concern itself with the higher security mechanisms, such as SSL. IPSec can be implemented in addition to such protocols.
- ❑ IPSec can be implemented in tunnel mode or transport mode.
- ❑ In the tunnel mode, the entire IP datagram, including its original header, is encrypted by IPSec and a new IP header is added.
- ❑ In the transport mode, the IP datagram except its header is encrypted by IPSec.
- ❑ The tunnel mode creates a virtual tunnel between the two communicating computers (usually routers).



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.





You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

10. Explain how NAT works with an example.
11. Why do we need to record port numbers in NAT?
12. What is a VPN?
13. Explain the AH and ESP protocols.
14. Explain how audit logs work.
15. What is a honeypot?



## DESIGN/PROGRAMMING EXERCISES

1. Study at least one real-life firewall product. Study its features with reference to the theory introduced in this chapter.
2. Try to download a free home firewall. Which of its features are annoying at times? Why?
3. Can a firewall double up as an anti-virus product? Why?
4. Configure the rules of a packet filter.
5. Study how VPN is implemented in real life. Does it need digital certificates? Why?
6. Implement NAT in software as a dummy exercise in Java. This NAT software should detect if a packet is outgoing or incoming and route it appropriately.
7. Enhance the above solution to also take care of port numbers so as to allow multiple internal hosts communicate with a single external host.
8. Study any one instance of real-life audit records. Is it sufficient to provide intrusion detection information?
9. Would you like to enhance the audit records structure? How?
10. Would you consider IPSec as a replacement for SSL? Why?
11. Do you think it is easier to implement IPSec than SSL? Why?
12. Would you ever propose leased line as a better approach than VPN? Why?
13. Does a firewall always have to be a router? Why?
14. What does it take for a host to become a firewall?
15. What is the difference between software and hardware firewalls?

## Case Studies on Cryptography and Security



### 10.1 Introduction

In this appendix, we discuss a few interesting case studies, based on our earlier technical details. As we shall note, implementing cryptography and security is not so straightforward. One needs to consider a lot of practical issues and technical concerns when implementing security.

The organization of the chapter is done in a manner that should help classroom discussions. Each case study begins with a few points for classroom discussion, which should augment the detailed discussion of the case study itself.

The coverage of case studies is very broad in nature. We have discussed diverse areas, such as cryptographic toolkits, Single Sign On (SSO), network-layer attacks, financial transaction security and a few cryptographic problems, which are very interesting to solve.

### 10.2 Cryptographic Solutions—A Case Study

#### *Points for classroom discussions*

1. *What are the key security aspects of a transaction?*
2. *How can the client side on the Internet have cryptographic capabilities?*
3. *What are cryptographic toolkits?*
4. *What are digital certificates, what is their use and how can their validity be checked?*

**Functional and Technical Requirements** Our case study deals with a banking application that is expected to provide cryptographic functionalities. For simplicity, we have kept the actual business logic quite straightforward, which may not be the case in real life. However, because the focus here is on the cryptographic functionalities, we shall treat the business requirements as ancillary.

Let us consider the requirements statement first.

Our cryptographic application would run on top of an existing banking application, wherein the end customers can perform a single task: funds transfer. A customer can transfer funds from her bank

account to any other person's account with the same bank. We shall consider the Internet (i.e. HTTP protocol) as the mechanism of communication between the client and the server. The following cryptographic services are required, depending on the amount of funds being transferred, as shown in Table 10.1.

**Table 10.1** *Application Requirements Depending on the Transaction Amount*

<i>Funds Transfer Amount</i>	<i>Cryptographic Functionality Required</i>
1-2000	<i>Message digest</i> – To verify the <i>finger print</i> of the transaction
2001-5000	<i>Digital signature</i> – To ensure message integrity and non-repudiation
5001 and above	<i>Digital signature</i> – To ensure message integrity and non-repudiation <i>Encryption</i> – To ensure confidentiality

Let us understand these requirements.

- As we can see, if the funds transfer transaction amount is up to 2000, we simply require a message digest to obtain and verify the *finger print* or integrity of the message. We assume the use of SSL, so that an attacker cannot alter both the message and the message digest. This is an example of *cryptography services*.
- If the transaction amount is between 2000 and 5000, we require a digital signature to ensure not only message integrity, but also non-repudiation. This is an example of *authorization services*.
- Finally, if the transaction amount exceeds 5000, we must not only sign a message but also encrypt it. This is a combination of *authorization services* and *cryptography services*.

Moreover, the authentication mechanism used by our application would be certificate-based authentication. That is, the user must sign a random challenge coming from the server with her private key and send it back to the server for verification, as a part of the authentication process.

Finally, in order to validate the user's certificate, we must provide for Certificate Revocation Lists (CRL) and Online Certificate Validation Protocol (OCSP) services.

We can broadly classify these requirements into appropriate categories as shown in Fig. 10.1.

**Hardware and Software Requirements** We have two clear portions of the application: the client-side and the server-side.

- **Client-side requirements** The client-side requirements would be a browser-based workstation that has Internet Explorer browser installed. The requirement for a specific browser is because we shall use the services of Microsoft's MS-CAPI cryptographic toolkit on the client-side. Since MS-CAPI is installed automatically as a part of Internet Explorer, we shall require that the user must have it installed. Of course, the user is free to use another browser (such as Netscape Communicator) for actual surfing/making transactions. We are simply saying that the user's workstation should still have Internet Explorer installed.
- **Server-side requirements** The server-side requirements will mainly consist of the presence of a cryptographic toolkit. Although JCA/JCE, MS-CAPI etc. can suffice, we would like to use a specialist toolkit such as the one from RSA, Entrust or Baltimore. There are no other special requirements on the server-side.

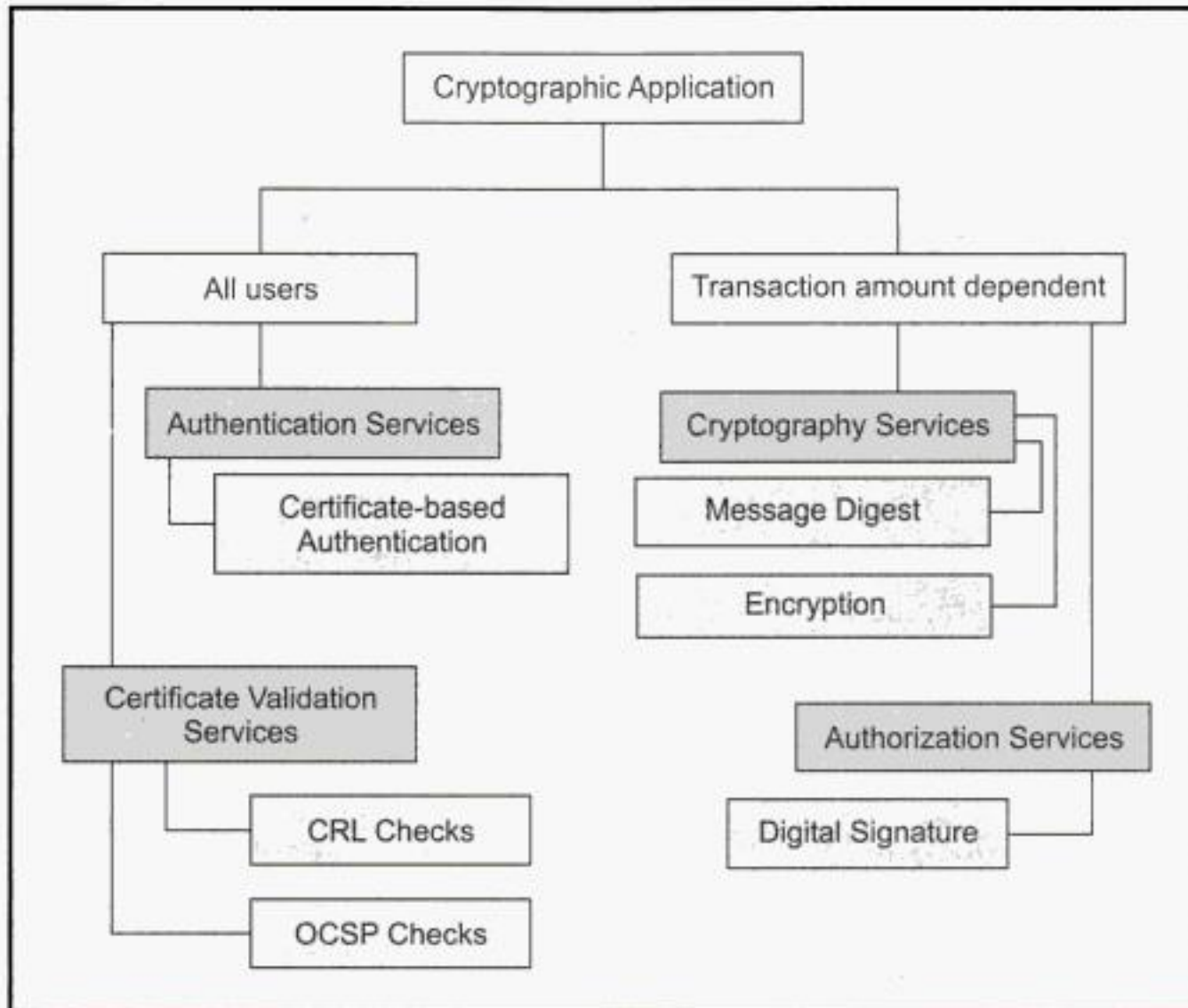


Fig. 10.1 Requirements classified into various cryptographic services

**Data Definitions** Let us now think what data storage requirements our application demands. We certainly need to modify the *User* table (used in chapter 8) to now add the user's digital certificate details in the place of the password field. These details would be necessary for our cryptographic functionalities. Portion of a sample *User* table is shown in Table 10.2.

Table 10.2 Portion of the User Table

Column Name	Data type and Size	Comments
User_Id	Char (10)	Not null, Primary key
First_Name	Char (30)	
Last_Name	Char (30)	
...	...	
Certificate	Binary	

Similarly, we need to create a new table for storing the random challenge during certificate-based authentication. Do not worry if you do not understand this. We shall revisit it soon. This *Session* table would look as shown in Table 10.3.

**Application Architecture** Let us now think about the architecture of our application. This involves thinking about the portions that are required on the client-side and those required on the server-side.

**Table 10.3** *New Session Table*

<i>Column Name</i>	<i>Data type and Size</i>	<i>Comments</i>
User_Id	Char (10)	Not null, Primary key
Random_Challenge	VarChar (30)	

Clearly, the user would perform the cryptographic operations such as message digests, digital signatures and encryption on the client-side (i.e. on the browser machine). We must verify the results of these operations on the server-side, such as checking if a digital signature was correct or to try and decrypt an encrypted message.

**Client-side Cryptographic Processing** On the client-side (i.e. the browser side), we have decided to use MS-CAPI for cryptographic functionalities. In order to make use of the MS-CAPI services, pure HTML pages are not sufficient. HTML pages (i.e. a Web page written in the HTML language) can be used for displaying text on the browser in a desired format. However, it cannot perform any client-side processing. Therefore, we must have some other mechanisms to utilize the services of MS-CAPI on the client-side.

As we know, the most widely used approaches for client-side programming are: client-side scripting (e.g. JavaScript or VBScript), Java applets or browser plug-ins (ActiveX controls). Unfortunately, JavaScript cannot directly access MS-CAPI. Therefore, that option is ruled out. We must now decide between applets and plug-ins. Applets are safer to use. But the drawback with applets is that they are quite slow, both because they are written in Java and also because they must be downloaded every time from the Web server to the browser when a cryptographic operation is to be performed. Consequently, we shall go for the approach of browser plug-ins.

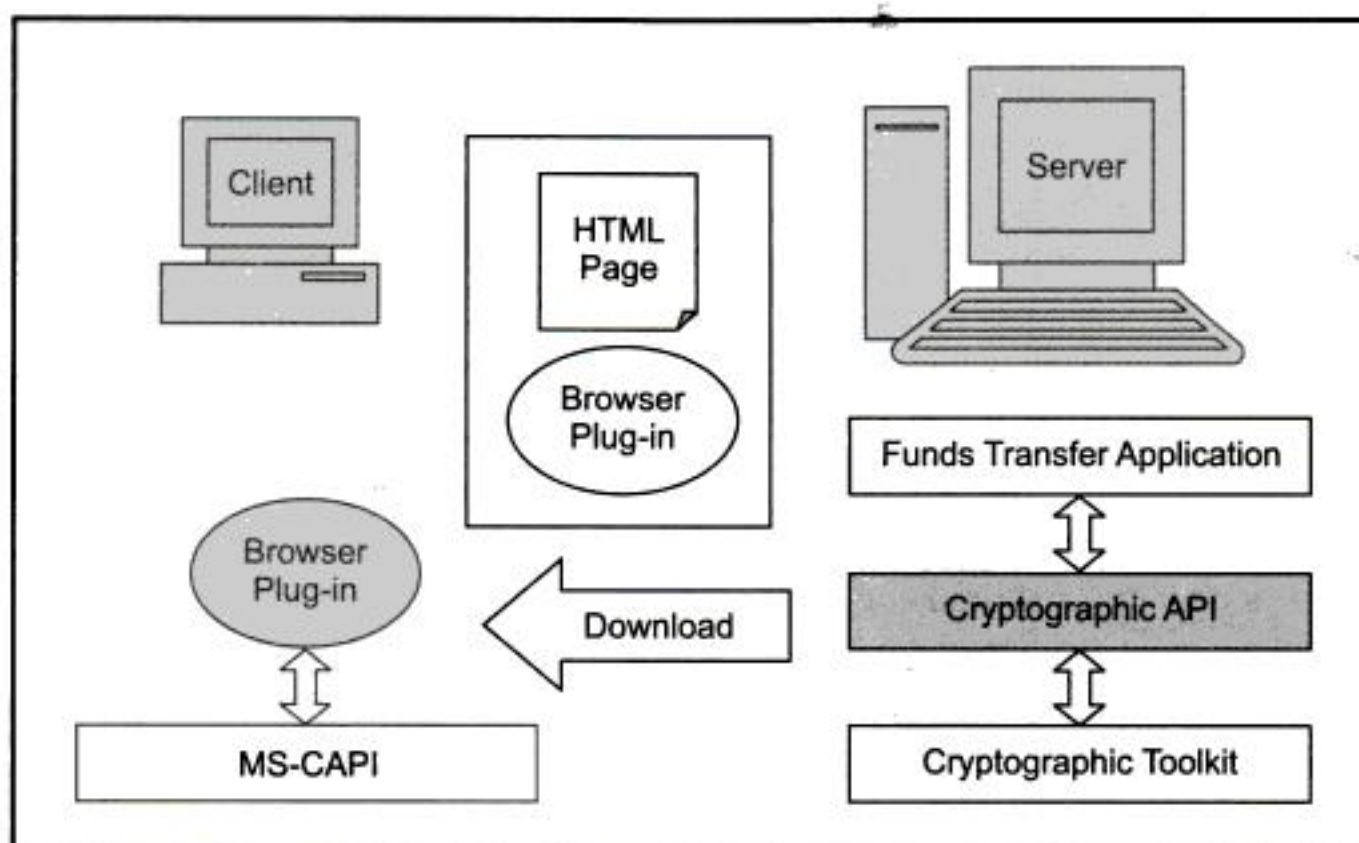
A plug-in is downloaded only once – for the very first time it is referenced in a Web page. After that, it resides inside the browser and can be reused directly without requiring a fresh download. Plug-ins do create some psychological doubts in the mind of the users, such as the possibility of performing malicious operations on the user's computer (e.g. introducing a virus or overwriting the disk contents). However, to guard against such doubts, we shall digitally sign the plug-ins, to create confidence in the mind of the end users. When a signed plug-in is downloaded from the Web server to the browser, the user is shown a message that the plug-in is signed and whether the user wants to trust the organization that has signed the plug-in.

**Server-side Cryptographic Processing** Server-side cryptographic functionalities are pretty much straightforward. We shall provide a dedicated cryptographic API, which can perform the cryptographic operations. The business application of funds transfer would need to call the functions provided by our cryptographic API. The cryptographic API, in turn, will call the appropriate toolkit API.

Thus, the overall cryptographic operations can be summarized as shown in Fig. 10.2.

In order to understand how this works, we shall consider all the desired cryptographic functionalities one-by-one, in turn.

**Message Digest** As we know, the creation of message digest involves the usage of an algorithm such as MD5 or SHA-1. We will provide a method *create\_digest ( )* in the plug-in, which, in turn, will call the message digest function provided by MS-CAPI. Let us understand how this works, step-by-step.



┆ Fig. 10.2 Conceptual view of the cryptographic operations on the server and the client

1. The user receives a screen for funds transfer, as shown in Fig. 10.3. The user has to enter the *From*, *To* account numbers and the amount to be transferred. Of course, in real life, the account numbers could be displayed in the form of a list from which the user can choose the account numbers. However, for simplicity, we shall ignore this possibility.

The screenshot shows a window titled "Funds Transfer". It contains three input fields: "From Account", "To Account", and "Amount". Below these fields are two buttons: "Ok" and "Cancel".

┆ Fig. 10.3 Funds transfer screen

2. Now, the actual message digest creation process will work as follows.
  - (a) When the user enters these details and presses the *Ok* button, a JavaScript function, which would be a part of the HTML page just displayed (i.e. *Funds Transfer*), would check the amount field. If the amount is less than or equal to 2000 (which is what we will



4. Based on the result of the toolkit operation, the *verify\_signature* ( ) method will send an appropriate message back to the calling method.

**Encryption** The encryption operation can be described as follows:

1. The input screen shown to the user will be the same as it was shown earlier. The user would enter the two account numbers and the amount and press the *Ok* button.
2. The JavaScript will sense that the amount is above 5000 and therefore, will call a plug-in method, such as *encrypt* ( ).
3. Our specifications suggest that we should first perform a digital signature and then encrypt the original data and the signature. Therefore, the *encrypt* ( ) method of the plug-in will first perform a digital signature, as described earlier, by calling the *digital\_signature* ( ) method.
4. After the signature is performed, we now wish to encrypt the original data and the signature together. Actually, this is enveloping and not encryption, as we shall see. For this purpose, the *encrypt* ( ) method of the plug-in will pass the original data, the signature and the server's public key to the appropriate MS-CAPI method.
5. The MS-CAPI method will automatically generate a one time symmetric key. It will encrypt the original data and the signature together with this symmetric key. It will then encrypt the symmetric key with the server's public key. This forms the digital envelope, which it returns back to the *encrypt* ( ) method of the plug-in.
6. The plug-in passes the digital envelope to the JavaScript, which submits the form to the server.

On the server-side, the following operations take place.

1. The server-side application will call the appropriate method supplied by our cryptographic API, such as *decrypt* ( ). It will also pass the digital envelope to the *decrypt* ( ) method.
2. The *decrypt* ( ) method will extract the server's private key from a pre-defined location and pass that along with the digital envelope to the underlying toolkit's appropriate method, such as *open\_envelope* ( ).
3. The toolkit method will open the envelope using the server's private key and obtain the one-time symmetric key. Using this symmetric key, it will decrypt the inside data block, to obtain the original data and the digital signature. It will return these values to the *decrypt* ( ) method.
4. The *decrypt* ( ) method has to now verify the signature. For this, it will call the *verify\_signature* ( ) method, which we have discussed earlier.
5. Based on the result of the de-signing operation, the *verify\_signature* ( ) method will send an appropriate message back to the *decrypt* ( ) method.
6. Depending on the return value of the *verify\_signature* ( ) method, the *decrypt* ( ) method will send an appropriate message back to the calling method.

**Certificate-based Authentication** We have discussed the concept of certificate-based authentication in great detail earlier. Therefore, we shall not repeat the discussion here. However, one point must be noted. We have created an additional *Session* table here. This table will be used to store the random challenge created by the server to verify the user's proof of possession of the digital certificate. For the sake of completeness, let us quickly revisit the steps involved in this process.

1. The user submits a login request to the server, only containing the user id.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

General Bank Of India (GBI) has implemented an Electronic Payment System called as *EPS* in about 1200 branches across the country. This system transfers payment instructions between two computerized branches of GBI. A central server is maintained at the EPS office located in Mumbai. The branch offices connect to the Local VSAT of a private network by using dial-up connection. The local VSAT has a connectivity established with the EPS office. GBI utilizes its proprietary messaging service called as *GBI-Transfer* to exchange payment instructions.

Currently, EPS has minimal data security. As the system operates in a closed network, the current security infrastructure may suffice the need. The data moving across the network is in encrypted format.

**Current EPS Architecture** EPS is used to transmit payment details from the payer branch to the payee branch via the central server in Mumbai. Fig. 10.5 depicts the flow, which is also described step-by-step.

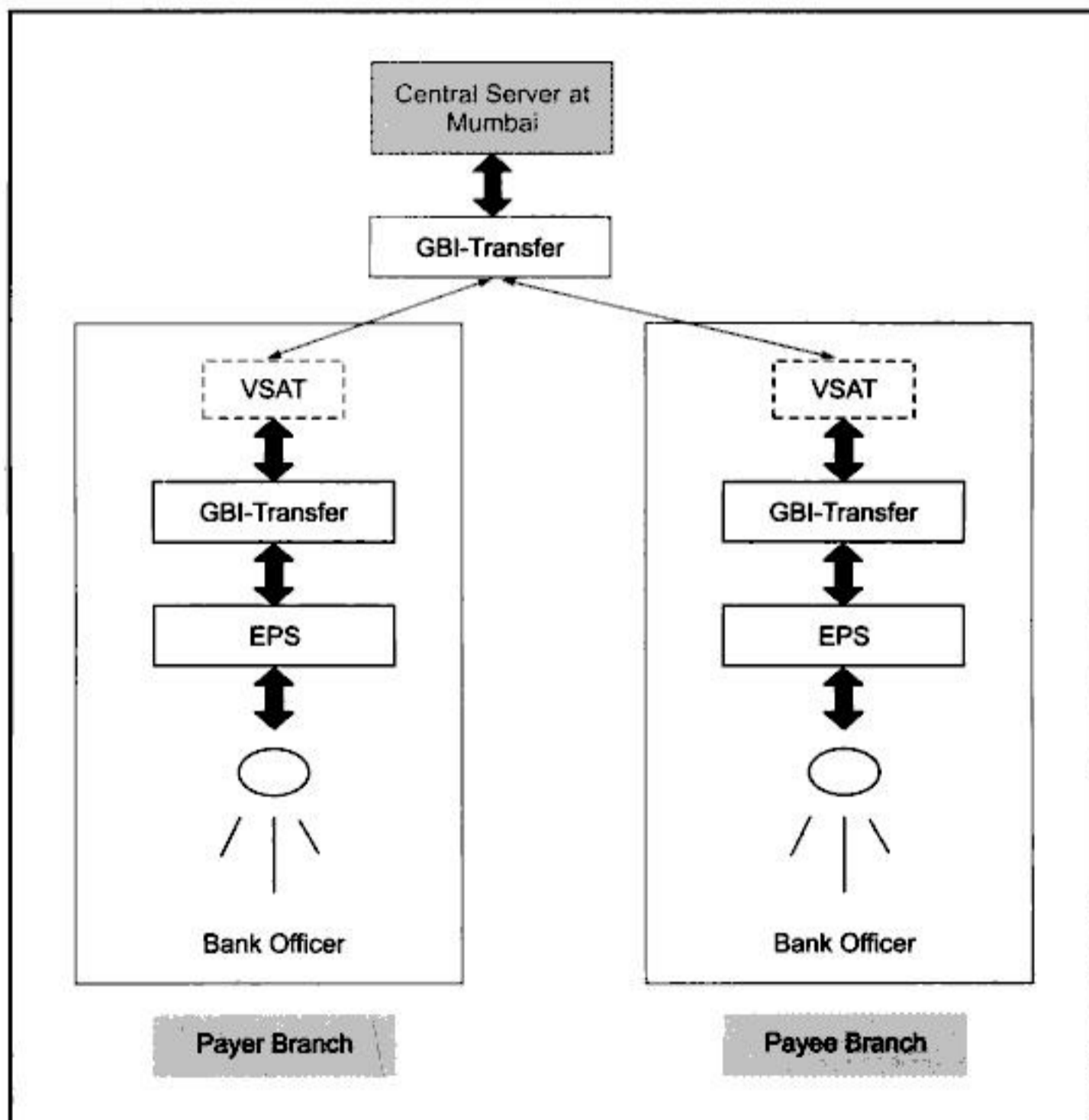


Fig. 10.5 EPS transaction flow

A typical payment transfer takes the following steps:

1. A data-entry person in the *Payer Branch* enters transaction details through the EPS interface.
2. A Bank Officer checks the validity of the transaction through the EPS interface.
3. After validating the transaction, the Bank Officer authorizes the transaction. Authorized transaction is stored in a local Payment Master (PM) database.
4. Once the transaction is stored in PM, a copy of the same is encrypted and stored in a file. This transaction file is stored in OUT directory.
5. The *GBI-Transfer* application looks for any pending transactions (i.e. for the presence of any files in the OUT directory) by a polling mechanism and if it finds such transactions, it sends all these files one-by-one to the EPS central office located in Mumbai by dialing the local VSAT.
6. The local VSAT gets connectivity to the EPS central office and the transaction is transferred and stored in the IN directory at the EPS central office.
7. The interface program at the EPS central office collects the file pending in the IN directory and sends it to the PM application at that office.
8. In order to send the Credit Request to PM, the transaction headers are changed. The transaction with changed headers in encrypted format is then placed in OUT directory of the EPS central office.
9. The GBI-Transfer application at the EPS central office collects the transactions pending in the OUT directory and sends them to the *Payee Bank* through the VSAT.
10. The transaction is transferred and stored in the IN directory of the *Payee Branch*.
11. The interface program at the *Payee Branch* collects the transaction and posts it in PM.
12. PM marks the credit entry and returns back an acknowledgement of the same. The acknowledgement is placed in OUT directory of the *Payee Branch*.
13. The acknowledgement is picked by GBI-Transfer at the *Payee Branch* and sent to the EPS central office through the VSAT.
14. The EPS central office receives the credit acknowledgement and forwards it to *Payer Branch*.
15. The *Payer Branch* receives the credit acknowledgement receipt. This completes the transaction.

**Requirements to Enhance EPS** As GBI is in the process of complete automation and setting up connectivity over the Internet or a private network, they need to ensure stringent security measures, which demand the usage of a Public Key Infrastructure (PKI) framework.

As a part of implementing security, GBI wants the following aspects to be ensured:

- Non-repudiation (Digital Signatures)
- Encryption – 128-bit (Upgrade to the current 56-bit encryption)
- Smart card support for storing sensitive data & on-card digital signing
- Closed loop Public Key Infrastructure

**Proposed Solution** Since providing cryptographic functionalities require the usage of a cryptographic toolkit, it is assumed that GBI will implement an appropriate Certification Authority (CA) infrastructure and a PKI infrastructure offering.

The transaction will be digitally signed and encrypted/decrypted at the Payer and Payee branches, as well as at the EPS central office. The signing operation can be performed on the system or on external

of the same is sent to EPS central Office. The Credit Response to the EPS central office can also be digitally signed and encrypted in a similar fashion.



## 10.5 Denial Of Service (DOS) Attacks

*Points for classroom discussions*

1. What is the Denial Of Service (DOS) attack?
2. Which of the DOS attacks – the one launched using TCP protocol or the one launched using the UDP protocol, more severe? Why?
3. What is the meaning of the term 'service' in DOS?
4. What can possibly prevent DOS attacks?

The **Denial Of Service (DOS)** attack has gained a lot of attention in the last few years. Most notable DOS attacks were launched against famous Web sites such as Yahoo, Amazon, etc. In April 2000, a 15-year old Canadian called as Mafiaboy launched DOS attacks, which caused a lot of concern regarding the security mechanisms of Web sites.

The basic purpose of a DOS attack is simply to flood/overhaul a network so as to deny the authentic users services of the network. A DOS attack can be launched in many ways. The end result is the flooding of a network or change in the configurations of routers on the network.

The reason it is not easy to detect a DOS attack is because there is nothing apparent to suggest that a user is launching a DOS attack and is actually not a legitimate user of the system. This is because in a DOS attack, the attacker simply goes on sending a flood of packets to the server/network being attacked. It is up to the server to detect that certain packets are from an attacker and not from a legitimate user and take an appropriate action. This is not an easy task. Failing this, the server would fall short of resources (memory, network connections, etc. and come to a grinding halt after a while.

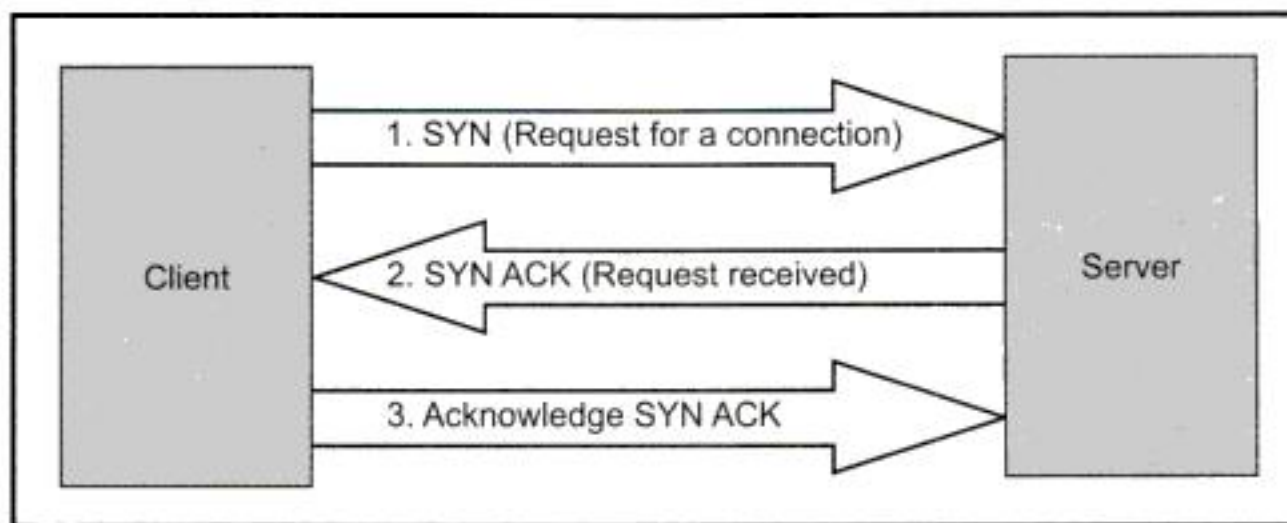
A typical mechanism to launch a DOS attack is with the help of the SYN requests. On the Internet, a client and a server communicate using the TCP/IP protocol. This involves the creation of a TCP connection between the client and the server, before they can exchange any data. The sequence of these interactions is as follows:

1. The client sends a SYN request to the server. A SYN (abbreviation of *synchronize*) request indicates to the server that the client is requesting for a TCP connection with it.
2. The server responds back to the client with an acknowledgement, which is technically called as SYN ACK.
3. The client is then expected to acknowledge the server's SYN ACK.

This is shown in Fig. 10.8.

Only after all the three steps above are completed that a TCP connection between a client and a server is considered as established. At this juncture, they can start exchanging the actual application data.

An attacker interested in launching a DOS attack on a server performs Step 1. The server performs Step 2. However, the attacker does not perform Step 3. This means that the TCP connection is not complete. As a result, the server needs to keep the entry for the connection request from the client as *incomplete* and must wait for a response (i.e. Step 3) from the client. The client (i.e. the attacker) is not at all interested in executing step 3. Instead, she simply keeps quiet. Now, imagine that the client sends



— Fig. 10.8 TCP connection establishment process

many such SYN requests to the same server and does not perform Step 3 in any of the requests. Clearly, a lot of incomplete SYN requests would be pending in the server's memory and if these are too many, the server could come to a halt!

Another mechanism to launch the SYN attack is, in Step 1, the attacker forges the source address. That is, the attacker puts the source address as the address of a non-existing client. Therefore, when the server executes Step 2, the SYN ACK never reaches any client at all, fooling the server!

In a more severe case, the attacker launches a Distributed DOS (DDOS) attack. Here, the attacker sends many SYN requests to the server from many physically different client computers. Thus, even if the server detects a DOS attack, it cannot do much by blocking SYN requests coming from a particular IP address – there are many such requests from a variety of (forged) clients!

Mafiaboy performed an attack, which was quite similar to what we have discussed here. He flooded the Web servers with packets sent one after the other in quick succession. This brought a few sites to halt for more than three hours. To stop these attacks, the site administrators started blocking packets coming from suspected source IP addresses. After a few hours, the attacks seemed to be nullified. What had actually happened was quite different. Mafiaboy had simply stopped his attacks! Perhaps he was too tired or bored!

How does one prevent DOS attacks? There are no clear answers here. The following mechanisms can be tried out:

1. Investigate the incoming packets and look for a particular pattern. If such a pattern emerges, then try blocking incoming packets from the concerned IP addresses. However, this is not easy as we have mentioned and this is even tougher to be performed in real time.
2. Another mechanism is to configure the services offered by a particular application so that it never accepts more than a particular number of requests in a specified time interval. Therefore, even if the attacker keeps sending more and more requests, the server would process them at its own pace and use that time to analyze/detect the attacks and take a corrective action.
3. Blocking a particular IP address, port number or a combination of such factors can also prevent a DOS attack. Of course, doing this in real time is not easy.
4. As a precaution, it is always good to have a backup of the firewall and the servers ready. If the main machine is compromised, it should be quickly brought down and the backup can take its place until a proper cleanup is performed.



## 10.6 IP Spoofing Attacks

*Points for classroom discussions*

1. *What is the IP spoofing attack?*
2. *Why is it not easy to detect IP spoofing attacks?*

**IP Spoofing** attacks are more challenging than the DOS attacks, if the attacker intends to use the consequence of the attack for his own benefit. Kevin Mitnick launched such an attack successfully on Tsutomu Shimomura's home computer and the computer at the University of Southern California. Before describing the attack, the technical mechanism to achieve it is first described as follows:

1. The attacker creates an IP datagram (packet) to be sent to the server, just like any other normal packet. This is a SYN request.
2. The main difference between the packet created by the attacker and any other packet is that the attacker puts the *source address* as another computer's IP address. That is, the attacker *forges* or *spoofs* the source IP address.
3. As usual, the server responds back with a SYN ACK response, which travels to the computer with the forged IP address (i.e. not to the attacker).
4. The attacker has to somehow get hold of this SYN ACK response sent by the server and acknowledge it, so as to complete a connection with the server.
5. Once this is done, the attacker can try various commands on the server computer.

Kevin Metnick was a person who had a lot of *background* in hacking/attacking computer systems. Following are some of the incidents he was involved in.

- (a) In his early days, Kevin joined a group of hackers. Among their many attacks, the most notable was the one in which they changed a home phone to a pay phone! Thus, when the phone subscriber wanted to make a call, a message greeted her with a request to first pay 20 cents!
- (b) At the age of 17, Kevin and his friend actually entered a phone company's office and stole passwords from there! This resulted into a jail term for them.
- (c) In 1983, Kevin tried to break into a Pentagon computer over the ARPAnet, an act that was detected, causing another imprisonment for Kevin.

There were many such examples.

In the IP spoofing attack, Kevin performed the following tasks in a very intelligent fashion. Before we explain it, it should be noted that Tsutomu Shimomura had a trusted relationship between his home computer (X) and the computers at the University of Southern California (Y).

1. Kevin first flooded Tsutomu's home computer (X) with a series of SYN requests, causing it to virtually come to a halt.
2. Kevin then sent a SYN request to the main server (Y) at the University. In this packet, Kevin put the source address as X. That is, the source address was spoofed.
3. The server (Y) detected this as an attempt to establish a request for a TCP connection and responded back with a SYN ACK response. As expected, the SYN ACK response went back to Tsutomu's home computer (X), because that was the source address in the original SYN request of Step 2.

4. Kevin had flooded X in Step 1. So, X is not able to see Y's response.
5. Kevin now guessed the sequence number used by Y in the SYN ACK response (after a few failures) and used that number in the acknowledgement of the SYN ACK message, which it sent to Y. That is, Kevin sent many acknowledgements (with different sequence numbers) of the SYN ACK message from Y, to Y.
6. In each case, Kevin immediately sent a command to add a wildcard entry to the trust relationship file maintained by Y, which made Y to trust anybody and everybody. Obviously, this command failed for all the unsuccessful attempts of Kevin to send an acknowledgement of the SYN ACK message. But it succeeded for the one acknowledgement, which succeeded. This opened up Y for all outside users!



## 10.7 Cross Site Scripting Vulnerability (CSSV)

Points for classroom discussions

1. *What is the purpose of scripting technologies on the Internet?*
2. *What can prevent CSSV attacks?*
3. *What sort of testing can the creators of a Web site perform in order to guard against possible CSSV attacks?*

**Cross Site Scripting Vulnerability (CSSV)** is a relatively new form of attacks that exploits inadequate validations on the server-side. The term *Cross Server Scripting Vulnerability (CSSV)* is actually not completely correct. However, this term was coined when the problem was not completely understood and has stuck ever since. Cross-site scripting happens when malicious tags and/or scripts attack a Web browser via another site's dynamically generated Web pages. The attacker's target is not a Website, but rather its users (i.e. clients or browsers).

The idea of CSSV is quite simple to understand and is based on exploiting the scripting technologies, such as JavaScript, VBScript or JScript. Let us understand how this works. Consider the following Web page containing a form as shown in Fig. 10.9, in which the user is expected to enter her postal address. Suppose that the URL of the site sending this page is `www.test.com` and when the user submits this form, it would be processed by a server-side program called as `address.asp`.

We would typically expect the user to enter the house number, street name, city, postal code and country, etc. However, imagine that the user enters the following weird string, instead:

```
<SCRIPT>Hello World</SCRIPT>
```

As a result, the URL submitted would be something like `www.test.com/address.asp?address=<SCRIPT>Hello World </SCRIPT>`.

Now suppose that the server-side program `address.asp` does not validate the input sent by the user and simply sends the value of the field `address` to the next Web page. What would this translate to? It would mean that the next Web page would receive the value of `address` as `<SCRIPT>Hello World</SCRIPT>`.

As we know, this would most likely treat the value of the `address` field as a script, which would be executed as if it is written in a scripting language, such as JavaScript etc on the Web browser. Therefore, the user would get to see *Hello World*.

 **10.8 Contract Signing**

*Points for classroom discussions*

1. *When can a contract in real-life considered to be complete?*
2. *Is it sufficient if only one of the parties signs a contract?*
3. *What mechanism can be used in cryptography to sign electronic contracts?*
4. *How can disputes be settled if a party later refuses having signed a contract?*

There is a builder, Bob, who has constructed a building for residential purposes. This building contains 20 flats (apartments). Bob wants to sell all of them to individual customers. Alice is one such customer, who is interested in buying a flat. She approaches Bob over telephone and expresses her desire to buy the flat. After a couple of such discussions, Alice decides to go ahead with her purchase. However, Bob has only one requirement. This requirement deals with the way the contract or the agreement between Bob and Alice is signed. Since Bob is Internet-savvy, he wants that the contract between him and Alice be digitally signed over the Internet and that Alice deposit the money into his account by using Internet banking.

In this discussion, we shall restrict our scope to only the first aspect, i.e. digitally signing contracts. How would Bob ensure that the contract signing process is complete in all respects? How would Alice be sure that she is not being cheated? How would a dispute be settled, if it arises?


In order to ensure that the contract signing happens smoothly, Bob contacts a respected third party, Trent. Alice also speaks with Trent over phone and is assured that Trent is a responsible third party, in which she could trust. With these ideas in place, let us write down the steps that would ensure that the digital contract signing is complete and comprehensive in all respects.

1. Alice digitally signs a copy of the contract and sends it to Trent over the Internet.
2. Bob digitally signs a copy of the contract and sends it to Trent over the Internet.
3. Trent sends a message to both Alice and Bob, informing them that he has received the digitally signed contracts from both.
4. At this stage, Alice digitally signs two more copies of the contract and sends both of them to Bob.
5. Bob now digitally signs both the contract copies received from Alice. He keeps one of the copies for his records and sends the other one to Alice.
6. Alice and Bob both inform Trent that they have a copy of the contract, which is digitally signed by both of them.
7. Trent now destroys the original contract copies received from Alice and Bob in Steps 1 and 2.

Is this solution foolproof? Let us examine it.

Can Alice deny at a later date that she never signed the contract? She cannot, because Bob has a copy of the contract, which was signed by him as well as by Alice. It is also the case the other way round. Bob cannot refute having signed the contract, because Alice has a copy of the contract signed by both.

Thus, the solution is indeed comprehensive. This solution also involves the use of a third party (also called as an *arbitrator*), Trent.

 **10.9 Secret Splitting**

*Points for classroom discussions*

1. *What is the need of secret splitting?*
2. *What would happen if we split a secret and one of the persons knowing the secret leaves the organization? Can the original secret still be recovered? Can the person leaving the organization break the secret before leaving?*
3. *Is the concept of secret splitting the same as XML signatures (signing only a portion of a document)?*

Many times, it is important to *split* a secret in such a fashion that the individual pieces of the secret make no sense. For example, suppose that in a bank, the account details are maintained in a database, which is protected by two *different* passwords. The two passwords are distinct and are known to two different officers. None of them knows both the passwords. Each one knows only her password. This ensures that to access the whole set of accounts, both the officers have to enter their passwords independently and only then the database can be accessed. This sort of scheme is actually used in many banks and other institutions as well.

Let us now think how a variation of this scheme can be implemented using cryptography. Suppose that Alice and Bob are the two officers and Trent is an arbitrator. The scheme should be devised in such a way that neither Alice nor Bob has an access to the complete secret, but they can combine their secrets so as to come up with the required combined secret. The scheme works as follows:

1. Suppose that the plain text secret (e.g., a combined password, which needs to be split into two parts) is  $P$ .
2. Trent generates a random number  $R$ , whose length in bits is exactly the same as that of  $P$ .
3. Trent performs an XOR operation on  $P$  and  $R$  to generate the combined secret,  $S$ , as follows:

$$S = P \oplus R$$

4. Trent now sends  $S$  to Alice and  $R$  to Bob.

This process is shown in Fig. 10.10.

The only way for Alice and Bob to regenerate the original secret  $P$ , is to perform the following operation:

$$P = S \oplus R$$

Alice cannot do this alone; because she does not have  $R$ . Bob cannot do it either, as it does not know  $S$ . The only way to regenerate  $P$  is for Alice and Bob to come together, enter their respective pieces of the secret (i.e.  $S$  and  $R$ , respectively), perform an XOR operation and generate  $P$ .



## 10.10 Virtual Elections

*Points for classroom discussions*

1. *Is it technically possible to have elections on the Internet? How? What sort of infrastructure would be needed for this?*
2. *What would be the main concerns in such a virtual election?*
3. *What would be the use of digital signatures and encryption in virtual elections?*

Another situation where cryptography is useful is virtual elections. Computerized voting would become quite common in the next few decades. As such, it is important that the protocol for virtual



## 10.12 Creating a VPN

*Points for classroom discussions*

1. Review the concepts of a VPN.
2. Discuss the pre-requisites of a VPN.
3. Create a VPN and test it, as explained below.

There are many companies/Web sites that offer free VPN software. An example is <http://sunsite.dk/vpnd>. This VPN software runs on Linux operating system. For the purpose of setting up the VPN, proceed as follows:

1. Use five layer-2 hubs, two routers (R1 and R2), two PCs running Linux (P1 and P2), which function as routers and five general purpose computers to form the intranet shown in Fig. 10.11.

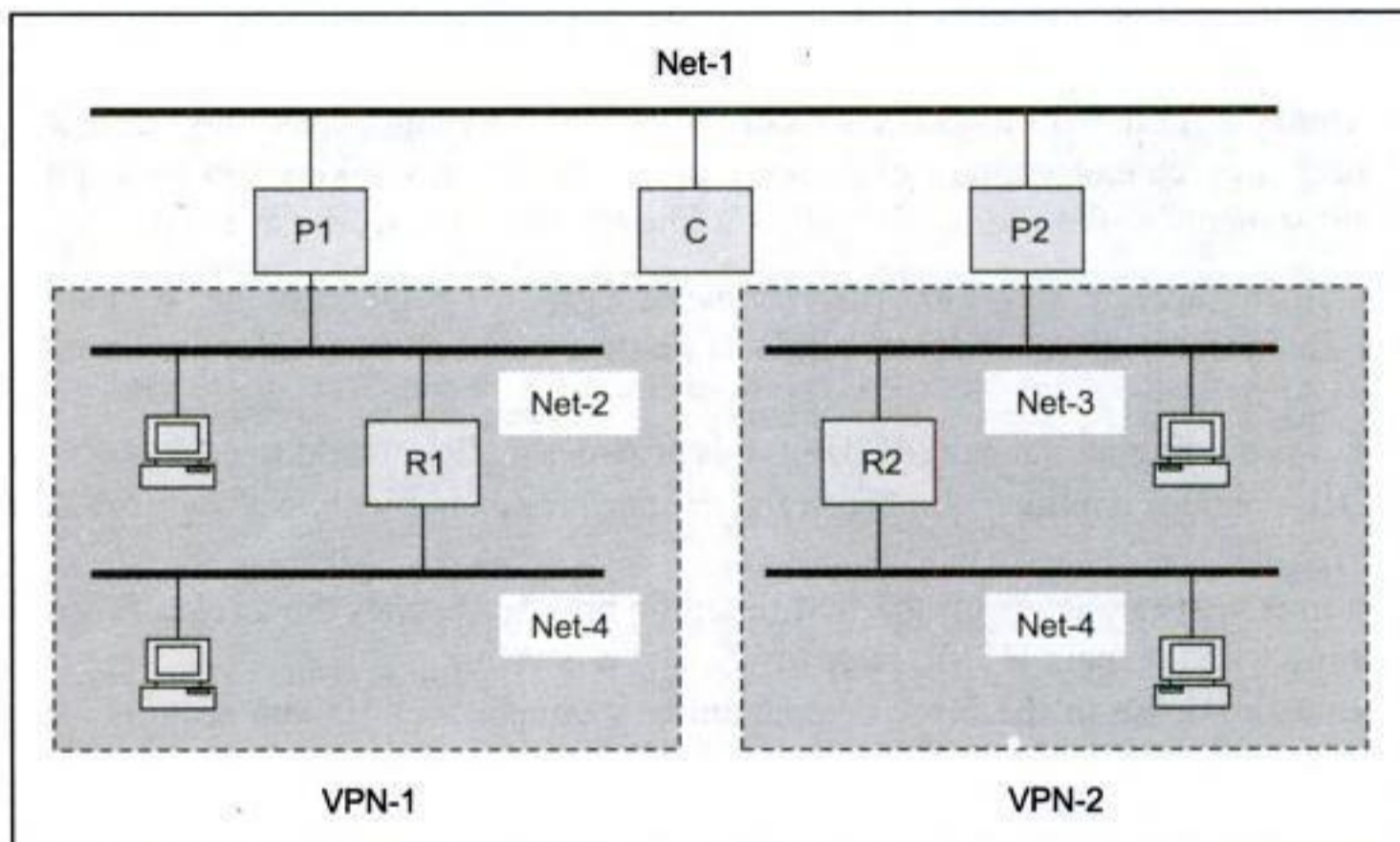


Fig. 10.11 VPN set up

2. Download and compile a VPN software for Linux and install it on the two Linux computers (P1 and P2).
3. Configure the VPN software so that it encrypts all the communication between the two computers.
4. Run software on the *checker computer* (C), which will capture and display all packets traveling on network 1.
5. Do a *TELNET* from a computer in location 1 on to another computer in location 2. Verify that all data is encrypted.

# Mathematical Background



## A.1 Introduction

Some readers may like to know the mathematical background behind the various cryptography techniques. We outline a few key concepts that are essential for this purpose. However, we must point out that this is not mandatory for a reader who is not keen to study these details, instead, is content with the conceptual view of cryptography.

## A.2 Prime Numbers

### A.2.1 Factoring

**Prime numbers** are very important in cryptography. A prime number is a positive integer, greater than 1, whose only factors are 1 and itself. That is, a prime number cannot be divided by any number other than 1 and itself. It should be obvious that 2, 3, 5, 7, 11, ... are prime numbers and 4, 6, 8, 10, 12, ... are not. There are an infinite number of prime numbers. Cryptography uses prime numbers heavily. Especially public key cryptography has its roots in prime number theory.

Figure A.1 shows a Java program for finding out whether any given integer is prime or not.

The reader is encouraged to modify the above program so as to automatically test for numbers from 2 to 1000 for primality. That is, the program should run a loop to test for all numbers between 2 and 1000 as to whether they are prime or not.

Two numbers are **relatively prime** when they have no factors in common other than 1. If the Greatest Common Divisor (GCD) of  $a$  and  $n$  is 1, it is written as  $\text{gcd}(a, n) = 1$ . As we will note, the numbers 21 and 44 are relatively prime (because they have no factors in common), but the numbers 21 and 45 are not (because they have a factor 3 in common).

### A.2.2 Euclid's Algorithm

A method of calculating the GCD of two numbers is by using the **Euclid's algorithm**. Let us write the C language representation of this algorithm, as shown in Fig. A.2.

```

// Java program to test whether a given number is prime
// Author: Atul Kahate

public class PrimeTest {
    public static void main (String [] args) {
        int numberToTest = 101;
        int m = 0;

        if (numberToTest <= 1) {
            System.out.println ("The number " + numberToTest + " is NOT prime");
            return;
        }

        for (int i=2; i<numberToTest - 1; i++) {
            m = numberToTest % i;
            if (m == 0) {
                System.out.println ("The number " + numberToTest + " is NOT prime");
                return;
            }
        }

        System.out.println ("The number " + numberToTest + " IS prime");
    }
}

```

**Fig. A.1** Java program to test whether a number is prime or not

```

int gcd (int x, int y)
{
    int a;

    /* If the numers are negative, make them positive */
    if (x < 0)
        x = -x;
    if (y < 0)
        y = -y;

    /* No point going ahead if the sum of the numbers is 0 */
    if ((x + y) = 0)
    {
        printf ("The sum of %d and %d is 0. ERROR!", x, y);
        return -1;
    }

    a = y;
    while (x > 0)
    {
        a = x;
        x = y % x;
        y = a;
    }

    return a;
}

```

**Fig. A.2** C language representation of the Euclid's algorithm

Let us trace the algorithm for  $x = 21$  and  $y = 45$ . The trace is shown in Figure A.3.

x	y	A
21	45	NA
3	21	21
0	3	3

Fig. A.3 Trace of the Euclid's algorithm

As we can see,  $\gcd(21, 45) = 3$  from the above table. By our earlier definition, therefore, 21 and 45 are not relatively prime.

### A.2.3 Modular Arithmetic and Discrete Logarithms

**Modular arithmetic** is based on simple principles. *Modulo* is the remainder left after an integer division. For example,  $23 \bmod 11 = 12$ , because 12 is the remainder of the division  $23 / 11$ . Modular arithmetic then says that 23 and 11 are equivalent. That is,  $23 \equiv 11 \pmod{12}$ . In general,  $a \equiv b \pmod{n}$  if  $a = b + kn$  for some integer  $k$ . If  $a > 0$  and  $0 < b < n$ , then  $b$  is the remainder of the division  $a / n$ . Other names for these are: **residue** for  $b$  and **congruent** for  $a$ . The triple equal to sign ( $\equiv$ ) denotes **congruence**. Cryptography uses computation  $\bmod n$  very frequently.

Modular exponentiation is a one-way function used in cryptography. Solving it is easy. For example, consider  $a^x \pmod{n}$ , given the values of  $a$ ,  $x$  and  $n$ . It is quite simple to solve. However, the inverse problem of modular exponentiation is that of finding the discrete logarithm of a number. This is quite tough. For instance, find  $x$  where  $a^x \equiv b \pmod{n}$ . As an example, if  $3^x \equiv 15 \pmod{17}$ , then  $x = 6$ . For large numbers, solving this equation is quite difficult.

### A.2.4 Testing for Primality

If  $p$  is an odd prime number, then the equation  $x^2 \equiv 1 \pmod{p}$  has only two possible solutions,  $x \equiv 1$  and  $x \equiv -1$ . We shall not discuss the proof of this.

### A.2.5 Square Roots Modulo a Prime

If  $n$  is the result of the multiplication of two prime numbers, then the ability to find out the square root  $\bmod n$  is equivalent to the ability of factoring  $n$ . That is, if we know the prime factors of  $n$ , then we can easily calculate the square roots of a number  $\bmod n$ .

### A.2.6 Quadratic Residues

If  $p$  is prime and  $0 < a < p$ , then  $a$  is a **quadratic residue**  $\bmod p$ , if:

$$x^2 \equiv a \pmod{p} \text{ for some } x$$

For instance, if  $p = 7$ , the quadratic residues are 1, 2 and 4, because:

$$1^2 = 1 \equiv 1 \pmod{7}$$

$$2^2 = 4 \equiv 4 \pmod{7}$$

$$3^2 = 9 \equiv 2 \pmod{7}$$

$$4^2 = 16 \equiv 2 \pmod{7}$$

$$5^2 = 25 \equiv 4 \pmod{7}$$

$$6^2 = 36 \equiv 1 \pmod{7}$$



## A.3 Fermat's Theorem and Euler's Theorem

Two theorems that are significant to public key cryptography are **Fermat's Theorem** and **Euler's Theorem**.

### A.3.1 Fermat's Theorem

This theorem states the following:

If  $p$  is prime and  $a$  is a positive integer not divisible by  $p$ , then we have:

$$a^{p-1} \equiv 1 \pmod{p}$$

Let us have  $a = 3$  and  $p = 5$ . Then, as per the above theorem, we have:

$$3^{5-1} \equiv 3^4 = 81 \equiv 1 \pmod{5}$$

Hence the proof.

Another form of this theorem states that if  $p$  is prime and  $a$  is any positive integer, then the following equation holds:

$$a^p \equiv a \pmod{p}$$

Let us consider a few examples:

1. Let  $a = 3$  and  $p = 5$ , then we have:

- (i)  $a^p = 3^5 = 243$ . Now, if we compute  $243 \pmod{5}$ , we will have a result of 3.
  - (ii)  $a \pmod{p} = 3 \pmod{5} = 3$ .
- Hence, we have:  $3^5 \equiv 3 \pmod{5}$ .

2. Let  $a = 4$  and  $p = 8$ , then we have:

- (i)  $a^p = 4^8 = 65536$ . Now, if we compute  $65536 \pmod{8}$ , we will have a result of 0.
- (ii)  $a \pmod{p} = 4 \pmod{8} = 4$ .

Now, the results of the above two steps are not matching. This is because  $p$  is not prime.

3. Let  $a = 4$  and  $p = 7$ , then we have:

- (i)  $a^p = 4^7 = 16384$ . Now, if we compute  $16384 \pmod{7}$ , we will have a result of 4.
  - (ii)  $a \pmod{p} = 4 \pmod{7} = 4$ .
- Hence, we have:  $4^7 \equiv 4 \pmod{7}$ .

### A.3.2 Euler's Theorem

Before discussing Euler's theorem, we need to discuss the **Euler Totient Function**. This function is written as  $j(n)$ , where  $j(n)$  is the number of positive integers less than  $n$  and relatively prime to  $n$ .

For instance, if  $n = 6$ , the positive integers less than  $n$  are 1, 2, 3, 4, and 5. Of these, only 1 and 5 do not have any factors common with 6. Thus,  $j(n) = j(6) = 2$ . Note that the number 6 is not prime. Let us consider a prime number  $n = 7$ . Here, all the positive integers preceding it (i.e. 1 to 6) are relatively prime to it. In general, for a prime  $n$ ,  $j(n) = (n - 1)$ .

Further, suppose  $p$  and  $q$  are two prime numbers. Then, for  $n = pq$ , we have:

$$j(n) = j(pq) = j(p) \times j(q) = (p-1) \times (q-1)$$

Let  $p = 3$ ,  $q = 7$ . Then,  $n = p \times q = 21$ .

Therefore,  $\phi(n) = j(21) = \phi(3) \times \phi(7) = 2 \times 6 = 12$ , where the 12 integers are 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20.

Based on this, Euler's theorem says that for every  $a$  and  $n$ , which are relatively prime, we have:

$$aj^{(n)} \equiv 1 \pmod{n}$$

Let  $a = 3$  and  $n = 10$ . Then, we have:

$j(n) = j(10) = 4$ , the 4 numbers being 1, 3, 7 and 9.

So,  $aj^{(n)} = 3^4 = 81 \equiv 1 \pmod{10}$ .



## A.4 Chinese Remainder Theorem

The **Chinese Remainder Theorem** uses the prime factorization of a number  $n$  to solve a system of equations. In general, if the prime factors of  $n$  are  $p_1 * p_2 * \dots * p_t$ , then, the system of equations

$$x \bmod p_i = a_i$$

where  $i = 1, 2, \dots, t$

has a unique solution  $x$ , provided  $x < n$ .

That is, for a number less than the product of a few primes, is uniquely identified by its residues moduls of those prime numbers.

For example, suppose we have two prime numbers as 5 and 7. Suppose that 16 is our number. Then, we have:

$$16 \bmod 5 = 1$$

$$16 \bmod 7 = 2$$

There is only one number smaller than  $5 * 7$ , i.e. 35 that has these residues: 16. These two residues can be used to uniquely determine the number. This can be proven by using the Java program shown in Fig. A.4.

```
// Java program to test Chinese remainder theorem basics
// Author: Atul Kahate

public class PrimeTest {
    public static void main (String [] args) {
        int k1 = 5, k2 = 7;

        for (int i=2; i<35; i++) {
            int n1 = i % k1;
            int n2 = i % k2;

            System.out.println ("Number = " + i + "Residues = " + n1 + " and " + n2);
        }
    }
}
```

**Fig. A.4** Chinese remainder theorem test

- (b) Multiply the cross-sectional values. (In this case, it means  $(18 \times 19)$  and  $(21 \times 2)$ ).
- (c) Subtract the results of these multiplications. (In this case, it means  $(18 \times 19) - (21 \times 2) = 300$ ).

When we compute the adjoint of other elements, the matrix looks as follows:

$$\begin{array}{ccc} +300 & -357 & +6 \\ -313 & +313 & +0 \\ +267 & -252 & -51 \end{array}$$

2. Now we need to transpose the matrix, that is, write columns as rows. Thus, values in column 1 of the matrix (i.e. +300, -313, and +267) will now become the values in row 1, and so on. The result looks as follows:

$$\begin{array}{ccc} +300 & -313 & +267 \\ -357 & +313 & -252 \\ +6 & +0 & -51 \end{array}$$

3. The determinant of the original matrix is -939. In Hill cipher, we need to take a mod 26 of this value, i.e.  $-939 \bmod 26 = -3$ . But we will disregard this possibility. Therefore, the inverse of the matrix is:

$$\begin{array}{ccc} 300/-939 & -313/-939 & 267/-939 \\ -357/-939 & 313/-939 & -252/-939 \\ 6/-939 & 0 & -51/-939 \end{array}$$

As another example, consider the following matrix:

$$\begin{array}{ccc} 20 & 15 & 18 \\ 78 & 95 & 56 \\ 43 & 89 & 32 \end{array}$$

Adjoint of this matrix is:

$$\begin{array}{ccc} -1944 & 1122 & -870 \\ -88 & -134 & 284 \\ 2857 & -1135 & 730 \end{array}$$

Determinant is 11226

Therefore, inverse is:

$$\begin{array}{ccc} -1944/11226 & 1122/11226 & -870/11226 \\ -88/11226 & -134/11226 & 284/11226 \\ 2857/11226 & -1135/11226 & 730/11226 \end{array}$$



## A.11 Mathematics behind Cryptographic Operation Modes (Reference: Chapter 3)

This section describes the mathematics behind the various modes of cryptographic operations, as described in Chapter 3 of this book.

- **Cipher Block Chaining (CBC) mode**

In the Cipher Block Chaining (CBC) mode, each cipher text block is passed through the decryption algorithm. The result then gets XORed with the previous cipher text block to yield the original plain text block. To have a look at the mathematics, let us have:

$$C_j = E_k [C_{j-1} \text{ XOR } P_j]$$

Based on this:

$$D_k [C_j] = D_k [E_k (C_{j-1} \text{ XOR } P_j)]$$

$$D_k [C_j] = C_{j-1} \text{ XOR } P_j$$

$$C_{j-1} \text{ XOR } D_k [C_j] = C_{j-1} \text{ XOR } C_{j-1} \text{ XOR } P_j = P_j$$

To produce the first block of cipher text, we know that an Initialization Vector (IV) is used, which is XORed with the first plain text block. Hence, for decryption, the first cipher text block is XORed with the IV to get the first block of plain text.

# Number Systems



## B.1 Introduction

A **number system** contains a set of numbers that have common characteristics. Let us discuss the common number systems that are used by humans (decimal) as well as computers (binary, octal, hexadecimal). In any number system, three aspects determine the value of each digit within a number:

1. The digit itself.
2. The position of the digit in that number.
3. The base of the number system.

The base of a number system is the number of different symbols used in that system. For example, in decimal number system the base is 10, because it uses 10 different symbols from 0 to 9.

## B.2 Decimal Number System

As we know, the decimal number system contains 10 different symbols, 0 to 9. Therefore, its base is 10. Thus, a number in the decimal number system is actually represented by multiplying each digit by its weight and then by adding all the products. For example, the value of a number 4510 in decimal number system is calculated as shown in Figure B.1.

Thousands position		Hundreds position		Tens position		Ones position		Total
$4 \times 10^3$	+	$5 \times 10^2$	+	$1 \times 10^1$	+	$0 \times 1^0$		--
4 x 1000	+	5 x 100	+	1 x 10	+	0 x 1		--
4000	+	500	+	10	+	0	=	4510

**Fig. B.1** Representation of a decimal number

### B.3 Binary Number System

The binary number system contains only two distinct symbols (called as binary digits or bits), 0 and 1. Therefore, its base is 2. We had used the increasing powers of 10 from right to left to represent decimal numbers. Here, we use increasing powers of 2. Thus, the value of a binary number 1001 in decimal is calculated as equal to 9 as shown in Figure B.2.

4 <sup>th</sup> position		3 <sup>rd</sup> position		2 <sup>nd</sup> position		1 <sup>st</sup> position		Total
$1 \times 2^3$	+	$0 \times 2^2$	+	$0 \times 2^1$	+	$1 \times 2^0$		--
$1 \times 8$	+	$0 \times 4$	+	$0 \times 2$	+	$1 \times 1$		--
8	+	0	+	0	+	0	=	9

Fig. B.2 Binary number representation

How can we convert a decimal number to its equivalent binary number? For that, we divide the number continuously by 2 till we get a quotient of 0. In every stage, we get 0 or 1 as the remainder. When we write these remainders in the reverse order, we get the equivalent binary number. This is as shown in Figure B.3, for a decimal number 500.

Divisor	Quotient	Remainder
2	500	
2	250	0
2	125	0
2	62	1
2	31	0
2	15	1
2	7	1
2	3	1
2	1	1
	0	1

Fig. B.3 Decimal to binary conversion

Here, the number 500 is divided continuously by 2, each time noting down the remainder (0 or 1). Finally, when the quotient is 0, we stop the process and write down the remainders in the reverse order. As we can see, the binary equivalent of a decimal number 500 is 111110100.

### B.4 Octal Number System

We know that the decimal number system has a base of 10 and that the binary number system has a base of 2. Similarly, the octal number system has a base of 8, as it represents 8 distinct symbols (0 to 7). The same principles, as discussed earlier are used for converting an octal number to decimal or vice versa.

The only change is, now 8 is used for multiplying weights or dividing quotients successively. So, a number 432 in octal can be converted into its decimal equivalent as shown in Fig. B.4.

3 <sup>rd</sup> position		2 <sup>nd</sup> position		1 <sup>st</sup> position		Total
$4 \times 8^2$	+	$3 \times 8^1$	+	$2 \times 8^0$		–
$4 \times 64$	+	$3 \times 8$	+	$2 \times 1$		--
256	+	24	+	2	=	282

Fig. B.4 Octal to decimal conversion

Let us work backwards and convert the decimal number 282 thus obtained to see if we get back the original octal number 432, as shown in Fig. B.5. Note that we go on dividing the decimal number continuously by 8 now, until we obtain a quotient of 0. In each case, we note the remainder and in the end, write all the remainders in the reverse order to get the equivalent octal number.

Divisor	Quotient	Remainder
8	282	
8	35	2
8	4	3
	0	4

Fig. B.5 Decimal to binary conversion

As the figure shows, the decimal number 282 is the same as octal number 432.



## B.5 Hexadecimal Number System

Hexadecimal number system is actually a superset of the decimal number system. We know that the decimal number system has 10 distinct symbols, from 0 to 9. Hexadecimal number system has all these 10 symbols and has 6 more (A through F). Thus, the hexadecimal number system contains 16 different symbols from 0 to 9 and then A through F. The symbol A in hexadecimal is equivalent to the decimal number 10, the symbol B is equivalent to the decimal number 11 and so on, which means that the last symbol F in hexadecimal number system is equivalent of the decimal number 15.

We follow the same conversion logic as we used for binary and octal number systems. The base is now 16. Let us convert a hexadecimal number 683C into its decimal equivalent, as shown in Fig. B.6.

4 <sup>th</sup> position		3 <sup>rd</sup> position		2 <sup>nd</sup> position		1 <sup>st</sup> position		Total
$6 \times 16^3$	+	$8 \times 16^2$	+	$3 \times 16^1$	+	$C \times 16^0$		–
$6 \times 4096$	+	$8 \times 256$	+	$3 \times 16$	+	$12 \times 1$		--
24576	+	2048	+	48	+	12	=	26684

Fig. B.6 Hexadecimal to decimal conversion

As before, let us convert this decimal number 26684 to see if we get back the hexadecimal number 683C using a decimal-to-hexadecimal conversion method, as shown in Fig. B.7.

Divisor	Quotient	Remainder
16	26684	
16	1667	C
16	104	3
16	6	8
	0	6

Fig. B.7 Decimal to hexadecimal conversion

We can see that we indeed obtain 683C in hexadecimal as equivalent of the decimal number 26684.



## B.6 Binary Number Representation

From the context of computers and data communications, the binary number system is most important, because it is used inside the computers to represent any alphabet, number or symbol. Therefore, let us discuss some aspects related to binary numbers.

• **Unsigned binary number** All binary numbers are unsigned by default, which means that they are positive. This is true in the case of decimal numbers as well. When we write a decimal number 457, we mean that it is implicitly (+)457. Let us now try to figure out the maximum number representation capabilities of the binary number system. As before, let us take the simple example of decimal number system.

Suppose we have a single slot, where we can store one decimal digit. How many different values it can store? Of course, it can store a digit between 0 and 9, thus 10 different values. Instead, if we decide to store a binary digit there, how many different values can it take? Of course, it can take a value of 0 or 1, thus 2 different values, where 1 is the maximum.

Now let us assume that we have two slots. If we decide to write decimal digits in each of them, then we can store a minimum number of 00 and a maximum number of 99 in decimal in these slots. Similarly, in binary, we can write 00 as the minimum number and 11 as the maximum number. Since binary 11 is 3 in decimal, the maximum number that we can in binary is actually decimal 3.

If we note, a pattern is emerging. Each new slot can accommodate the maximum digit in a given number system. Thus, if we have three slots, we can have the maximum decimal number as 999 and the maximum binary number as 111 (which is 7 in decimal).

Let us tabulate these results only for binary numbers, as shown in Fig. B.8. We shall continue the list, as we are observing some commonalities as shown in the last column.

The last column indicates how we can find out the maximum number possible in any storage allocated for binary numbers. It is simply 2 raised to the number of bit positions allocated minus 1. Thus, when we have 8 slots or 8 bit positions, we can represent decimal numbers from 0 (minimum number) to 255 (maximum number), a total of 256 different numbers. Since a computer *byte* generally contains 8 bits, a byte can represent one of the 256 distinct values (i.e. a number between 0 and 255).



### C.3 Perfect Secrecy

A cryptosystem wherein the cipher text gives absolutely no information about the plain text (except its size) achieves **perfect secrecy**. According to Shannon, this is possible only if the number of possible keys is greater than or equal to the number of possible messages. That is, the key must be equal to or longer than the message itself and no key should be reused. Thus, one time pad is the only candidate for perfect secrecy.

### C.4 Unicity Distance

**Unicity distance** is the approximation of the amount of cipher text so that the sum of the real information (entropy) in the corresponding text, plus the entropy of the encryption key equals the number of cipher text bits used. Furthermore, cipher texts longer than this distance are fairly certain to decrypt to only one plain text. On the other hand, cipher texts shorter than this distance generally have multiple, equally valid decryptions. Therefore, they are more secure, as the cryptanalyst has to pick up the correct one.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



# Index



- .NET cryptography [400-403](#)
- 1-factor authentication [357](#)
- 2-factor authentication [357](#)
- 3-D Secure Protocol [299-302](#)
- 3G security [332-335](#)
  
- A**
- Abstract Syntax Notation ([ASN.1](#)) [527](#)
- Access control [7](#), [10](#), [11](#)
- Access Control List (ACL) [10](#)
- Access right [410](#)
- Accessibility [11](#)
- Accuracy [11](#)
- Active attacks [14-15](#)
- Active Web page [267](#)
- ActiveX control [19](#)
- Advanced Encryption Standard (AES) [137-148](#)
- AES
  - See *Advanced Encryption Standard (AES)*
- Algorithm mode [87](#), [90](#), [98](#)
- Algorithm modes, Summary of [99](#)
- Algorithm type [87](#)
- Alteration of messages [15](#)
- Amount of information [521](#)
- Anomaly detection [475](#)
- Applet [19](#)
- Application gateway [441-442](#)
- Application level attacks [16](#)
- Assurance [11](#)
- Asymmetric key cryptography
  - History [153-154](#)
  - Overview [154-156](#)
- Asymmetric key cryptography [61-65](#), [72](#)
- Attacks, *security* [12](#)
- Attribute certificate [237](#)
- Audit
  - log [473](#)
  - record [473](#)
- Authentication [7](#), [8](#), [11](#)
- Authentication Header (AH) [455](#), [459-463](#)
- Authentication server [355](#)
- Authentication token [354-365](#)
- Authority Revocation List (ARL) [231](#)
- Automating attacks [3](#)
- Availability [7](#), [10](#), [11](#)
  
- B**
- Base CRL [228](#)
- Base-64 encoding [311](#), [312-314](#)
- Basic Encoding Rules (BER) [529](#)
- Behaviour-blocking [26](#)
- Bell-LaPadula Model [6](#)
- Biometric authentication [371-372](#)
- Birthday attack [171](#)
- Block cipher [88](#), [89-90](#)
- Blowfish [131-137](#)
- Book cipher [59](#)
- Boot sector virus [17](#)
- Brand theft [13](#)
- Brute-force attack [44](#)

Bucket brigade attack

See *Man-in-the-middle attack*

Buffer overflow attack 282-284

## C

Caesar cipher [41-43](#)

Canonical conversion 311

Certificate Management Protocol (CMP) 240

Certificate policies 240

Certificate Practice Statements (CPS) 240

Certificate revocation [226](#)

Certificate Revocation List (CRL) 227-231

Certificate Signing Request (CSR) 212

Certificate-based authentication 365-370

Certification Authority (CA) 207

Chain of trust 221

Challenge/response token 358

Chinese Remainder Theorem [511](#)

Chosen cipher text attack 80

Chosen text attack 80

Cipher Block Chaining (CBC) 91, 92-93

Cipher Feedback (CFB) 91, 93-95

Cipher text only attack 77-79

Circuit gateway [442](#)

Clandestine user [473](#)

Clear text [40](#)

Client-side script 23

Clogging attack [467](#)

Collision (*Message digest*) 170

Confidentiality 7, [8](#), [11](#)

Confusion 90

Congestion attack [467](#)

Congruence [509](#)

Cookie 21

Counter (CTR) 97

Criminal attacks [12](#)

Cross-certification 224

Cryptanalysis [38](#), [44](#)

Cryptanalyst 44

Cryptographic toolkits 403-404

Cryptography [38](#), [39](#)

## D

Data Certification Service (DCS) 240

Data Encryption Standard (DES) 100-115

Database control 409

Database security 409-426

Decryption 59

Decryption algorithm 60

Delta CRL 228

Demilitarized Zone (DMZ) 451

Denial of Service (DOS) [15](#), [16](#)

DES

See *Data Encryption Standard (DES)*

DES, Analyzing [110](#)

DES-2

See *Double DES*

DES-3

See *Triple DES*

Destruction [13](#)

Detection-specific audit record [473](#)

Dictionary attack 243

Differential cryptanalysis [110](#)

Diffie-Hellman key agreement 65-72, 206

Diffusion 90

Digital cash

See *Electronic money*

Digital

certificate 206

envelope 163

signature 165-197

Digital Signature

Algorithm (DSA) 194-197

Standard (DSS) 194

Discretionary control 409

Distinguished Encoding Rules (DER) 529

Distinguished Name (DN) 209

Distributed intrusion detection [475](#)

DNS spoofing

See *Pharming*

Domain Name System (DNS) 30

DOS

See *Denial of Service (DOS)*

Double DES [111-114](#)

Double spending problem [307](#)

Dual signature 292

Dynamic

packet filter 440

Web page 267

## E

Electronic cash

- See *Electronic money*  
 Electronic Code Book (ECB) 91-92  
 Electronic money  
   Introduction 302-303  
   Security in [303-304](#)  
   Types [304-306](#)  
 Electronic money  
 ElGamal 198-199  
 Elliptic curve 198  
 Elliptic Curve Cryptography (ECC) 198  
 Email security [307-326](#)  
 Encapsulating Security Payload (ESP) 455, 464-466  
 Encryption 59  
 Encryption algorithm 60  
 Entropy 521  
 Ethical and legal issues 11  
 Euclid's algorithm [507](#)  
 Euler Toient Function [510](#)  
 Euler's theorem [510](#)
- F**
- Fabrication 8  
 Factoring [507](#)  
 False Accept Ration (FAR) [371](#)  
 False Reject Ration (FRR) [371](#)  
 Fermat's theorem [510](#)  
 Fingerprint of a message  
   See *Message digest*  
 Firewall  
   Configurations [448-451](#)  
   DMZ 451  
   Introduction [435-437](#)  
   Limitations 451  
   NAT [443-448](#)  
   Types 437-443  
 Fraud [13](#)
- G**
- Group 90  
 GSM security 330-332
- H**
- Hasse's Theorem 512  
 Hill cipher 54  
 HMAC 190-194  
 Homophonic substitution cipher 46  
 Honeypot [475](#)  
 Host security 6  
 Hyper Text Markup Language (HTML) 23  
 Hyper Text Transfer Protocol (HTTP) 19
- I**
- IDEA  
   See *International Data Encryption Algorithm (IDEA)*  
 Identity theft [13](#)  
 Information Theory 521  
 Initialization Vector (IV) 93  
 Integrity 7, [8](#), [9](#), [11](#)  
 Intellectual property theft [13](#)  
 Interception [8](#), [13](#)  
 International Data Encryption Algorithm (IDEA)  
   115-123  
 Internet Key Exchange (IKE) 458  
 Internet Security Association and Key Management Protocol (ISAKMP) [467-469](#)  
 Interruption [14](#)  
 Intruder [472](#)  
 Intrusion [472](#)  
 Intrusion Detection System (IDS) 474  
 Intrusion prevention 474  
 IP Security (IPSec)  
   Introduction [452-454](#)  
   Key management [467-469](#)  
   Overview [454-455](#)  
   Protocols 455-466  
 ISAKMP/Oakley  
   See *Internet Security Association and Key Management Protocol (ISAKMP)*  
 Iteration count 243
- J**
- Jacobi symbol 512  
 Java applet  
   See *Applet*  
 Java application security 27  
 Java cryptography 393-400  
 Java Cryptography Architecture (JCA) 394  
 Java Cryptography Extensions (JCE) 394  
 Java keystore  
   See *Keystore*  
 Java keytool

See *Keytool*

## K

Kerberos 372-377  
Key 60  
Key archival 239  
Key Distribution Center (KDC) 378  
Key distribution, *problem of*  
    See *Key exchange, problem of*  
Key Encryption Key (KEK) 242  
Key exchange, *problem of* 62  
Key legitimacy (in PGP) [321](#)  
Key pair [72](#)  
Key range 73-77  
Key ring (in PGP) 317-318  
Key size  
    See *Key range*  
Key wrapping 162  
Keystore 252  
Keytool 252  
Knapsack  
    algorithm 197-198  
    problem 197-198  
Known plain text attack 79-80

## L

Lattice-based information 6  
Legal attacks [12](#)  
Legendre symbol 512  
Lightweight Directory Access Protocol (LDAP) 215  
Linear cryptanalysis [110](#)  
Login only 379  
Longitudinal Redundancy Check (LRC) 168

## M

Macro virus 18  
Mandatory control 409  
Man-in-the-middle attack 68-72  
Masquerader [15](#), [472](#)  
Massey-Omura Protocol 512  
Matrix inversion 513  
MD2 172  
MD3 173  
MD5 algorithm 172-182  
Meet-in-the-middle attack 112  
Memory-resident virus 17

Message Authentication Code (MAC) [188-189](#)  
Message digest 167-197  
Message digest of password 345  
Metamorphic virus 17  
MIME  
    See *Multipurpose Internet Mail Extensions (MIME)*  
Misfeasor [472](#)  
Modification [9](#), [14](#), [15](#)  
Modular arithmetic [509](#)  
Mono-alphabetic cipher 44-46  
Multi-factor authentication 357  
Multipurpose Internet Mail Extensions (MIME) [322](#)  
Mutual authentication 379

## N

Native audit record [473](#)  
Network Address Translation [443-448](#)  
Network level attacks 16  
Network security 6  
No security 6  
Non-repudiation 7, [9](#), [10](#), [11](#)  
Number systems [516-520](#)

## O

Object privilege 412  
OCSP request 232  
OCSP responder 232  
OCSP response 232  
Offline certificate revocation check 227  
One-time pad 58-59  
One-time password 355  
One-way authentication 379  
One-way public key 379  
Online certificate revocation check 232  
Online Certificate Status Protocol (OCSP) 232-235  
Operating systems security 404-409  
Orange book 6  
OSI Security Model  
    See *OSI Standard for Security Model*  
OSI Standard for Security Model [11](#)  
Output Feedback (OFB) 91, 95-96

## P

Packet filter 438-441  
Packet sniffing  
    See *Sniffing*

- Packet spoofing
    - See *Spoofing*
  - Parasitic virus 17
  - Passive attacks [14](#)
  - Passport 377
  - Password 341-354
  - Password Based Encryption (PBE) 242
  - Password encryption 352
  - PCMCIA cards 238
  - Penetration identification [475](#)
  - Perfect secrecy [522](#)
  - Person-in-the-middle attack
    - See *Man-in-the-middle attack*
  - PGP certificate [318-321](#)
  - Pharming 30-33
  - Phishing 29, 30
  - [PKCS#10](#) 245
  - [PKCS#11](#) 245
  - [PKCS#12](#) 245-246
  - [PKCS#14](#) 246-247
  - [PKCS#5](#) 242-244
  - [PKCS#8](#) 244
  - PKI
    - See *Public Key Infrastructure*
  - PKIX model 239
  - Plain text [40](#)
  - Playfair cipher 47-54
  - Polyalphabetic substitution cipher 47
  - Polygram substitution cipher 46
  - Polymorphic virus 17
  - Pretty Good Privacy (PGP) 314-322
  - Prime number [507](#)
  - Privacy [11](#)
    - concerns 3
  - Privacy Enhanced Mail (PEM) [310-314](#)
  - Private key [72](#)
  - Privilege 410
  - Product cipher [41](#)
  - Profile-based detection [475](#)
  - Property [11](#)
  - Proxy server
    - See *Application gateway*
  - Pseudo-random number generator 246
  - Public key [72](#)
  - Public key cryptography
    - See *Asymmetric key cryptography*
  - Public Key Cryptography Standard (PKCS) 212
  - PKCS standards 241-248
  - Public Key Infrastructure (PKI) 205
  - Publicity attacks [12](#)
- Q**
- Quadratic Reciprocity Theorem 512
  - Quadratic residue [509](#)
- R**
- Radix-64 encoding
    - See *Base-64 encoding*
  - Rail fence technique 54-55
  - Random challenge 348
  - Random Number Generator (RNG) 246
  - RC4 123-125
  - RC5 125-131
  - Reference monitor 4
  - Reflection attack 384
  - Registration Authority (RA) 209
  - Relatively prime number [507](#)
  - Release of message contents [14](#)
  - Replay attacks [15](#), 347
  - Rijndael
    - See *Advanced Encryption Standard (AES)*
  - Roaming certificate 235-237
  - Root CA 221
  - RSA algorithm 156-160
  - Rule-based detection 474
  - Running key cipher 59
- S**
- S/MIME
    - See *Secure MIME (S/MIME)*
  - Salt 243
  - Sandbox model 26
  - Scam [13](#)
  - Screened Host Firewall, Dual-Homed Bastion 450
  - Screened Host Firewall, Single-Homed Bastion 449
  - Secret key [72](#)
  - Secure DNS 33
  - Secure Electronic Transaction (SET)
    - and SSL [298-299](#)
    - Conclusion 297
    - Internals 289-297
    - Introduction 286-287

- Participants 287-288
  - Process 288-289
  - Secure Hash Algorithm (SHA)
    - See [SHA-1](#)
  - Secure Hyper Text Transfer Protocol (SHTTP) 284-285
  - Secure Mime (S/MIME) [322-327](#)
  - Secure Socket Layer (SSL)
    - Alert protocol 281-282
    - Handshake protocol 274-279
    - Introduction 272
    - Record protocol 279-281
    - Sub-protocols 274
  - Security Association (SA) 459
  - Security Association Database (SAD) 459
  - Security handshake pitfalls 379-387
  - Security management practices 7
  - Security models 6
  - Security policy 7
  - Security through obscurity 6
  - Security, *principles of* [7](#)
  - Seed 355
  - Self-signed certificate 223
  - SET
    - See *Secure Electronic Transaction (SET)*
  - [SHA-1](#) algorithm 182-185
  - SHA-512 185-188
  - Shared secret 379
  - SHTTP
    - See *Secure Hyper Text Transfer Protocol (SHTTP)*
  - Signed applet 21
  - Simple Certificate Validation Protocol (SCVP) 232-235
  - Simple columnar transposition technique 55-58
  - Simple Mail Transfer Protocol (SMTP) 308
  - Single Sign On (SSO) 387
  - Smart card [370-371](#)
  - SMTP
    - See *Simple Mail Transfer Protocol (SMTP)*
  - Sniffing 28, 29
  - Spoofing 28, 29
  - SSL
    - See *Secure Socket Layer*
  - SSO
    - See *Single Sign On (SSO)*
  - Stateful packet filter 440
  - Stateless (HTTP protocol) 21
  - Static Web page 265
  - Statistical anomaly detection 474
  - Statistical database 423
  - Stealth virus 17
  - Steganography 73
  - Stream cipher 88-89
  - Substitution [41](#)
  - Symmetric key cryptography 61-65, 98-100
  - Symmetric versus Asymmetric key cryptography 160-161
  - System privilege 412
- T**
- TCP/IP
    - See *Transmission Control Protocol/Internet Protocol (TCP/IP)*
  - TCP/IP vulnerabilities 406
  - Threshold detection [475](#)
  - Time Stamping Authority (TSA) 285
  - Time Stamping Protocol (TSP) 285-286
  - Time-based token 362
  - Timestamping service 240
  - Timing attack [111](#)
  - TLS
    - See *Transport Layer Security (TLS)*
  - Traffic analysis [14](#)
  - Transmission Control Protocol/Internet Protocol (TCP/IP) 270
  - Transport Layer Security (TLS) 284
  - Transport mode [456](#)
  - Transposition [41](#)
  - Triple DES 114-115
  - Trojan Horse 18-19
  - Trusted Computing Base (TCB) 6
  - Trusted system 4
  - Trusted third party [72](#)
  - Tunnel mode [456](#)
  - Types of attack, Summary of 81
- U**
- Unicity distance [522](#)
  - UNIX security 406-408
- V**
- Vernam cipher 58-59

Virtual Private Network (VPN) 469-472

Virus 16

Virus types 17

VPN

See *Virtual Private Network (VPN)*

## W

WAP gateway 327

WAP security 327-330

Web browser 19

Web of trust (in PGP) [321-322](#)

Web page 19

Windows security 408-409

Wireless Transport Layer Security (WTLS) 328-330

Worm 18

WTLS

See *Wireless Transport Layer Security (WTLS)*

## X

X.500 207

[X.509](#) 207

XML digital signature 249-251

XML encryption 248

XML Key Information Service Specifications (X-KISS) 252

XML Key Management Specifications (XKMS) 251-252

XML Key Registration Service Specifications (X-KRSS) 252

XML security 247-249

This book clearly explains the concepts in Cryptography and the principles employed behind Network Security. The text steers clear of complex mathematical treatment and presents the concepts involved through easy-to-follow examples and schematic diagrams.

**Salient features:**

- ✦ Uses a bottom-up approach.
- ✦ Stress on case studies and practical implementations of cryptography.
- ✦ Pedagogy includes
  - + 125 Design/Programming Exercises
  - + 140 Multiple Choice Questions
  - + Over 600 figures

**New to this edition:**

- ✦ Expansion of coverage on AES, PGP, S/MIME and Blowfish.
- ✦ New sections on Birthday Attacks, Trusted Systems, OSI Architecture, Buffer Overflow, and Legal/Ethical Issues have been added.
- ✦ Updated and expanded sections on
  - + .NET Cryptography
  - + TCP/IP Vulnerabilities
  - + Security in Operating Systems
  - + NAT (Network Address Translation)
  - + Audit Records
  - + Honeypots
  - + Java and Digital Certificates
- ✦ 138 new exercises have been added.

**Advance praise for the book:**

"...The writing style of the author is appreciably lucid...The table of contents is very rich and well designed... The book has a balanced coverage of both cryptography and security."

Visit <http://www.mhhe.com/kahate/cns2e> for supplements



Tata McGraw-Hill

Visit us at : [www.tatamcgrawhill.com](http://www.tatamcgrawhill.com)

ISBN-13: 978-0-07-064823-4

ISBN-10: 0-07-064823-9



9 780070 648234

Copyrighted material