

THE MANGA GUIDE™ TO

COMICS
INSIDE!

CRYPTOGRAPHY

MASAAKI MITANI
SHINICHI SATO
IDERO HINOKI
VERTE CORP.



OHM
Ohmsha

no starch
press

PRAISE FOR THE MANGA GUIDE SERIES

"Highly recommended."

—CHOICE MAGAZINE ON *THE MANGA GUIDE TO DATABASES*

"The *Manga Guides* definitely have a place on my bookshelf."

—SMITHSONIAN MAGAZINE

"The art is charming and the humor engaging. A fun and fairly painless lesson on what many consider to be a less-than-thrilling subject."

—SCHOOL LIBRARY JOURNAL ON *THE MANGA GUIDE TO STATISTICS*

"Stimulus for the next generation of scientists."

—SCIENTIFIC COMPUTING ON *THE MANGA GUIDE TO MOLECULAR BIOLOGY*

"The series is consistently good. A great way to introduce kids to the wonder and vastness of the cosmos."

—DISCOVERY.COM

"Absolutely amazing for teaching complex ideas and theories . . . excellent primers for serious study of physics topics."

—PHYSICS TODAY ON *THE MANGA GUIDE TO PHYSICS*

"A great fit of form and subject. Recommended."

—OTAKU USA MAGAZINE ON *THE MANGA GUIDE TO PHYSICS*

"I found the cartoon approach of this book so compelling and its story so endearing that I recommend that every teacher of introductory physics, in both high school and college, consider using it."

—AMERICAN JOURNAL OF PHYSICS ON *THE MANGA GUIDE TO PHYSICS*

"This is really what a good math text should be like. Unlike the majority of books on subjects like statistics, it doesn't just present the material as a dry series of pointless-seeming formulas. It presents statistics as something *fun* and something enlightening."

—GOOD MATH, BAD MATH ON *THE MANGA GUIDE TO STATISTICS*

"A single tortured cry will escape the lips of every thirty-something biochem major who sees *The Manga Guide to Molecular Biology*: 'Why, oh why couldn't this have been written when I was in college?'"

—THE SAN FRANCISCO EXAMINER



"A lot of fun to read. The interactions between the characters are lighthearted, and the whole setting has a sort of quirkiness about it that makes you keep reading just for the joy of it."

—HACKADAY ON *THE MANGA GUIDE TO ELECTRICITY*

"The *Manga Guide to Databases* was the most enjoyable tech book I've ever read."

—RIKKI KITE, LINUX PRO MAGAZINE

"*The Manga Guide to Electricity* makes accessible a very intimidating subject, letting the reader have fun while still delivering the goods."

—GEEKDAD

"If you want to introduce a subject that kids wouldn't normally be very interested in, give it an amusing storyline and wrap it in cartoons."

—MAKE ON *THE MANGA GUIDE TO STATISTICS*

"A clever blend that makes relativity easier to think about—even if you're no Einstein."

—STARDATE, UNIVERSITY OF TEXAS, ON *THE MANGA GUIDE TO RELATIVITY*

"This book does exactly what it is supposed to: offer a fun, interesting way to learn calculus concepts that would otherwise be extremely bland to memorize."

—DAILY TECH ON *THE MANGA GUIDE TO CALCULUS*

"Scientifically solid . . . entertainingly bizarre."

—CHAD ORZEL, SCIENCEBLOGS, ON *THE MANGA GUIDE TO RELATIVITY*

"Makes it possible for a 10-year-old to develop a decent working knowledge of a subject that sends most college students running for the hills."

—SKEPTICBLOG ON *THE MANGA GUIDE TO MOLECULAR BIOLOGY*

"*The Manga Guide to the Universe* does an excellent job of addressing some of the biggest science questions out there, exploring both the history of cosmology and the main riddles that still challenge physicists today."

—ABOUT.COM

"*The Manga Guide to Calculus* is an entertaining comic with colorful characters and a fun strategy to teach its readers calculus."

—DR. DOBB'S

THE MANGA GUIDE™ TO CRYPTOGRAPHY



THE MANGA GUIDE™ TO
CRYPTOGRAPHY

MASAAKI MITANI,
SHINICHI SATO,
IDERO HINOKI AND
VERTE CORP.



THE MANGA GUIDE TO CRYPTOGRAPHY.

Copyright © 2018 by Masaaki Mitani, Shinichi Sato, Idero Hinoki, and Verte Corp.

The Manga Guide to Cryptography is a translation of the Japanese original, *Manga de waku angou*, published by Ohmsha, Ltd. of Tokyo, Japan, © 2007 by Masaaki Mitani, Shinichi Sato, Idero Hinoki, and Verte Corp.

This English edition is co-published by No Starch Press, Inc. and Ohmsha, Ltd.

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-10: 1-59327-742-3

ISBN-13: 978-1-59327-742-0

Publisher: William Pollock

Production Editor: Laurel Chun

Authors: Masaaki Mitani and Shinichi Sato

Illustrator: Idero Hinoki

Producer: Verte Corp.

Developmental Editor: Jan Cash

Translators: Raechel Dumas and Asumi Shibata

Technical Reviewer: Erik Lopez

Additional Technical Reviewers: Jean-Philippe Aumasson and Ron Henry

Copyeditor: Paula L. Fleming

Compositors: Laurel Chun and David Van Ness

Proofreader: Lisa Devoto Farrell

Indexer: BIM Creatives, LLC

For information on distribution, translations, or bulk sales, please contact No Starch Press, Inc. directly:
No Starch Press, Inc.

245 8th Street, San Francisco, CA 94103

phone: 1.415.863.9900; info@nostarch.com; <http://www.nostarch.com/>

Library of Congress Cataloging-in-Publication Data

Names: Mitani, Masaaki, author. | Sato, Shinichi (Professor of engineering), author. | Hinoki, Idero, author. | Verte Corp., author.

Title: The manga guide to cryptography / Masaaki Mitani, Shinichi Sato, Idero Hinoki, and Verte Corp.

Other titles: Manga de waku angou. English

Description: San Francisco : No Starch Press, Inc., [2018] | Translation of
Manga de waku angou, published by Ohmsha, Ltd. of Tokyo, Japan, 2007.

Identifiers: LCCN 2018017805 (print) | LCCN 2018018420 (ebook) | ISBN
9781593278502 (epub) | ISBN 1593278500 (epub) | ISBN 9781593277420 (pbk.)
| ISBN 1593277423 (pbk.)

Subjects: LCSH: Public key cryptography--Comic books, strips, etc. |
Telecommunication--Security measures--Comic books, strips, etc. | Graphic
novels.

Classification: LCC TK5102.94 (ebook) | LCC TK5102.94 .M5813 2018 (print) |
DDC 652/.8--dc23

LC record available at <https://lccn.loc.gov/2018017805>

No Starch Press and the No Starch Press logo are registered trademarks of No Starch Press, Inc. Other product and company names mentioned herein may be the trademarks of their respective owners. Rather than use a trademark symbol with every occurrence of a trademarked name, we are using the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The information in this book is distributed on an "As Is" basis, without warranty. While every precaution has been taken in the preparation of this work, neither the authors nor No Starch Press, Inc. shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in it.

All characters in this publication are fictitious, and any resemblance to real persons, living or dead, is purely coincidental.

CONTENTS

PREFACE	xi
PROLOGUE	1
1	
THE FOUNDATIONS OF ENCRYPTION	15
What Is a Cipher?	17
Shannon's Encryption Model	20
The Relationship Between the Encryption Key and the Decryption Key	21
Classic Encryption Methods	24
Caesar Cipher	24
Substitution Cipher	25
Polyalphabetic Cipher	26
Transposition Cipher	27
Cipher Security	28
The Substitution Cipher's Key Space	31
Permutation, Combination, and Frequency Analysis	32
The Polyalphabetic Cipher's Key Space	33
The Transposition Cipher's Key Space	34
When Breaking Classical Encryption is Possible	37
Perfectly Secure Encryption	37
2	
SYMMETRIC-KEY ALGORITHMS	47
Binary Digits and Logical Operators	48
Symmetric-Key Algorithms	59
Stream Ciphers	65
Block Ciphers	68
CBC Mode	71
DES Ciphers	72
The Basic Configuration of a Feistel Cipher	73
Involution	74
Generating DES Encryption Subkeys	77
The DES Round Function f	78
The Full Structure of DES Encryption and Decryption	79
3-DES Encryption and AES Encryption	80
An Outline of AES	85
Simplified DES Encryption and Decryption	89

Converting Data into Binary	89
Generating a DES Ciphertext	90
Perform the Initial Permutation	91
Separate Data into Higher- and Lower-Order Bits	92
Feistel Rounds Variant	92
Swap the Higher- and Lower-Order Bits	95
Perform the Final Permutation	95
Decrypting a DES Ciphertext	97
Perform the Initial Permutation	97
Separate Data into Higher- and Lower-Order Bits	97
Round 1	97
Round 2	98
Generating DES Keys	100
Perform Permuted Choice PC-1	101
Rotate the Bits	102
Perform Permuted Choice PC-2	102
Rotate the Bits Again	103
Compress the Bits	103
How Reduced DES Differs from Real DES	104

3

PUBLIC-KEY ENCRYPTION	105
The Basics of Public-Key Encryption	106
Mathematical Dependencies of Public-Key Encryption	115
One-Way Functions	116
The Birth of RSA Encryption	119
Prime Numbers and Integer Factorization	120
Types of Primality Tests	129
Modulo Operations	134
Addition and Subtraction in Modulo Operations	137
Multiplication and Division of Modulo Operations	146
Fermat's Little Theorem and Euler's Theorem	152
Fermat: the Father of Number Theory	153
Fermat's Method and Pseudoprime Numbers	155
Euler's Theorem	156
The Mathematician Euler	157
Euler Function on the Product of Two Prime Numbers	158
The Structure of RSA Encryption	161
RSA Encryption: Encryption and Decryption	163
Generating an RSA Encryption Key	165
Verifying That the Public Key e and Private Key d Work in RSA Encryption	166
How to Make Public Keys and Private Keys	167
Generating a Ciphertext Using RSA	169
Decrypting a Ciphertext Using RSA	171

Public-Key Encryption and Discrete Logarithm Problems	173
Discrete Logarithm Problems	174
Encryption and Decryption in ElGamal Encryption	176
The Extended Euclidean Algorithm	181
Calculating the Private Key in RSA Key Generation	182
4	
PRACTICAL APPLICATIONS OF ENCRYPTION	185
Hybrid Encryption	186
Hash Functions and Message Authentication Codes	190
Falsification	190
Tampering Countermeasures	192
Hash Functions	193
Identity Fraud	194
Identity Fraud Countermeasures	195
Structure of Message Authentication Codes	196
Repudiation	197
Drawbacks of Message Authentication Codes	199
Digital Signatures	200
Repudiation Measures	200
The Workings of a Digital Signature	201
Man-in-the-Middle Attack	203
Using Certificates to Protect Against Man-in-the-Middle Attacks	204
Public-Key Infrastructure	206
Zero-Knowledge Interactive Proof	217
Preparatory Stage	217
Implementation Stage (Verification Process)	218
Methods of Identity Fraud	220
Anti-Identity Fraud in a Zero-Knowledge Interactive Proof	222
WHAT NEXT?	224
Pseudorandom Numbers and Encryption Security	224
PGP	224
SSL/TLS	225
Quantum Cryptography	225
Biometrics	225
INDEX	227

PREFACE

When thinking about information societies, which rely on networks that function with the internet at their core, we can see how many aspects of our lives have become more convenient. In addition to now being able to search for information on the web and communicate via email, other network services like online shopping and internet banking have become widespread.

While the age of networking comes with many benefits, we also encounter phrases like, “safe and secure,” “information security,” “protection of personal information,” “encryption,” and so on that have a somewhat unpleasant ring to them. These words are everywhere—not a day goes by without seeing them. Why is such talk so prevalent?

Networks require a variety of information to be exchanged, and this includes information that could cause problems when in the hands of others and information that we wish to keep secret. We must take measures to protect information in order to prevent important data such as our credit card numbers, bank account numbers, medical histories, loan debts, and email addresses from easily leaking to outsiders. This information can be abused, and as such there is no doubt that protecting information is the most important issue that faces us in the age of networking. In our connected society, where there are so many unreliable factors, the ability to distinguish authentic information and provide protection from dangers such as fraud, forgery, revision, and interception is crucial. *Cryptography* is the core technology we use to create network services that the public can use safely.

Cryptographic technology has evolved rapidly in recent years. It is no longer just the domain of information security specialists, and is now necessary knowledge for the users themselves, who up until now have used convenient network services without a thought.

So how then does cryptography work? How does it actually provide information security and protect personal information?

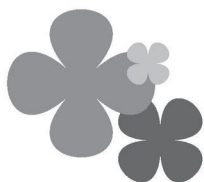
Using manga, this book explains how cryptography works and what it does. The complex math is broken down into friendly explanations. We’ve done our best to make this an effortless learning experience where you can enjoy the story. Of course, there are also ciphertexts worked into the story, so do your best to decipher them while having fun. By the time you’ve finished reading this book, you’ll no doubt have a thorough understanding of the fundamentals of cryptography and security.

Lastly, we’d like to express our most sincere gratitude to those who helped us publish this book, first to the folks at Ohmsha, Ltd. and also to Mr. Idero Hinoki who provided the illustrations.

MASAAKI MITANI AND SHINICHI SATO
APRIL, 2007

The page is framed by a wide border of repeating dark and light gray floral motifs. Inside this, a double-line rectangular frame is present, with small square notches at each corner. The word "PROLOGUE" is centered in the upper half of the page.

PROLOGUE



A CERTAIN CITY IN A CERTAIN
PREFECTURE, 78TH PRECINCT

SQUAD
ROOM

RECEPTION

RUKA
MEGURO

BIG BROTHER, BUY
IT FOR MEEEE!

INSPECTOR
JUN MEGURO

NO WAY! A HIGH
SCHOOL GIRL
DOESN'T NEED A
COMPUTER THAT
FANCY!

SLUUUUURP

$$ax + by = \gcd(a, b)$$

$$a^{p-1} \equiv 1 \pmod{p}$$

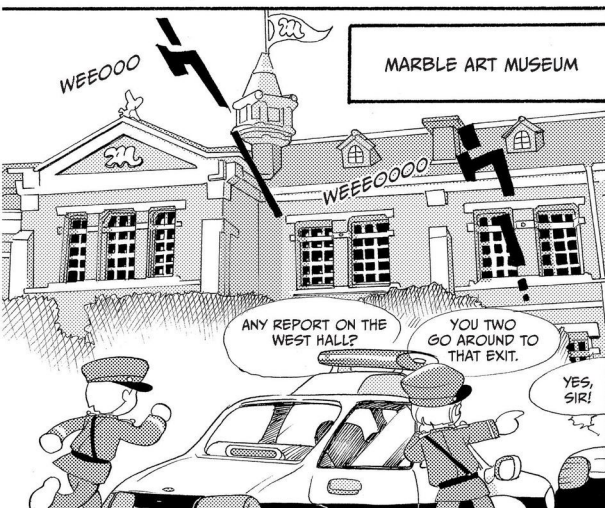
$$\varphi(p) = p - 1$$

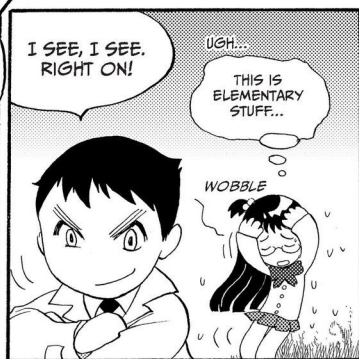
BUT I WANT TO
USE IT TO STUDY
MATHEMATICS!

BUT I WANT IT!

y^2

...





THE LOCATION OF THE
PAINTING DEPOSITORY IS
PROTECTED BY A CIPHER
TO KEEP UNAUTHORIZED
PEOPLE FROM FINDING IT.



THE DETAPOSITATORY IS
WATAREHOTAU SEFITAVE

ALL RIGHT! NO
WAY AN OUTSIDER
COULD HAVE
FIGURED THAT OUT!

THAT'S AWFUL.

← RUKA

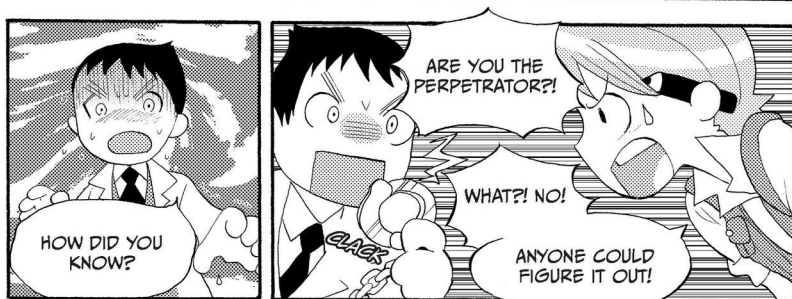
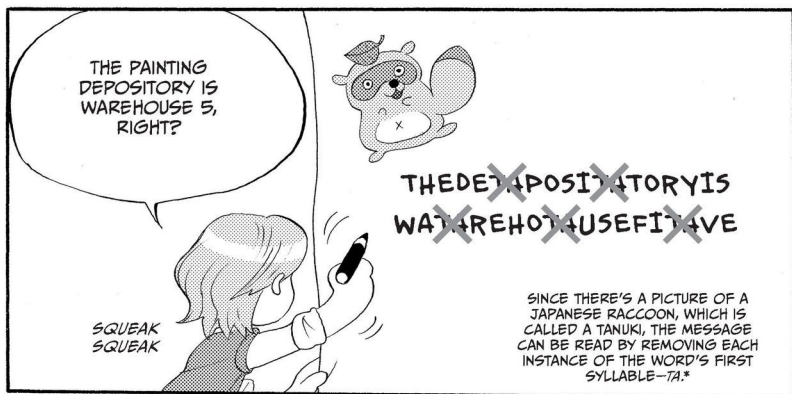
YOU CALL THAT
SECURITY?

CLICK

WHO ARE
YOU?!

CLICK
CLICK

RIO YONEDA, FROM
THE NIGHTLY NEWS!



* THE TANUKI CIPHER IS A COMMON WORDPLAY GAME IN JAPAN.
THE CIPHER IS BASED ON A PUN OF THE WORD TANUKI, WHICH
CAN MEAN "RACCOON" OR "OMIT THE TA'S."

WAIT, REALLY?



INSPECTOR MEGURO!
FORGET ABOUT HER.
FIND THE REAL THIEF—
AND QUICK!

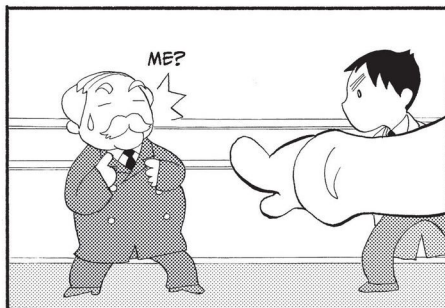
THE PERP...
THE PERP IS...



THE PERP IS
RIGHT THERE!

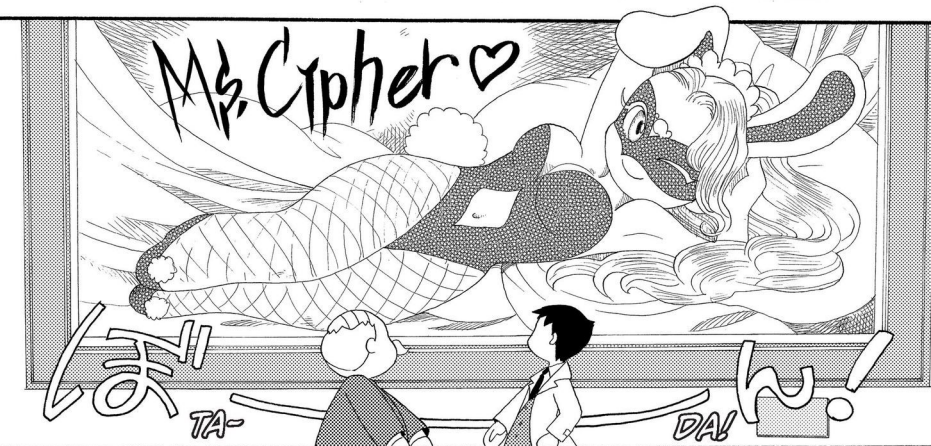


ME?



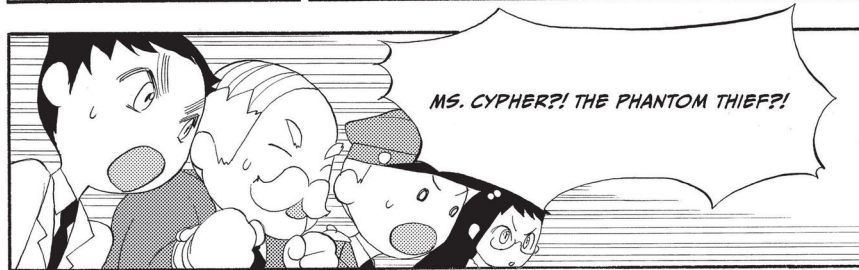
REALLY? YOU'RE
UNDER ARREST!







YOU'VE BEEN VISITED
BY MS. CYPHER.
I'VE TAKEN THE PAINTING.
NEXT I'LL TAKE VDVIRCU.
GOOD NIGHT.



I'VE TAKEN THE PAINTING.
NEXT I'LL TAKE VDVIRCU.

VDVIRCU—WHAT'S THAT?
IS THAT ENGLISH?

I
DON'T KNOW
ANY FOREIGN
LANGUAGES,
SO...

TEE-HEE

IT'S A CIPHERTEXT!
A CIPHERTEXT TELLING
US WHAT WILL BE
STOLEN NEXT!

UNLIKE WITH
THE MUSEUM'S
CIPHERTEXT, I'M
NOT FINDING ANY
MEANINGFUL WORDS
BY OMITTING
LETTERS...

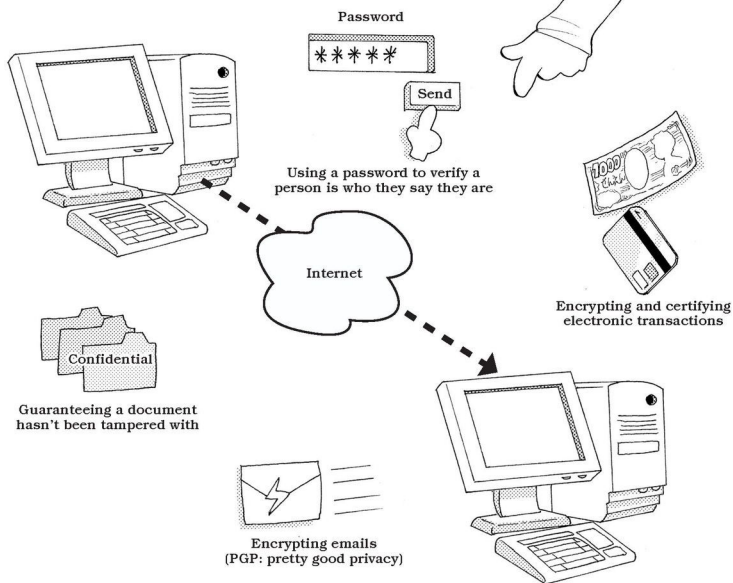
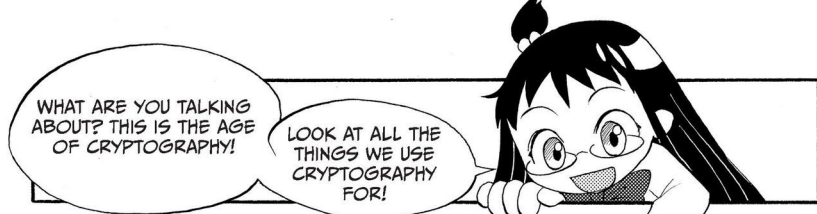
VDVIRCU

IN THAT CASE, LET'S
STUDY CRYPTOGRAPHY
AND TAKE DOWN
MS. CYPHER!

FIRE!

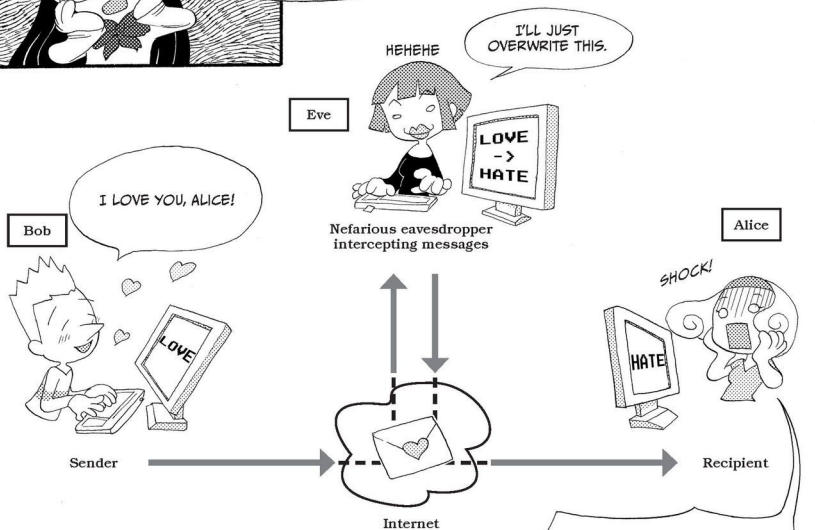
UP!

BUT THIS ISN'T SOME SPY
NOVEL. IS CRYPTOGRAPHY
REALLY THAT USEFUL?



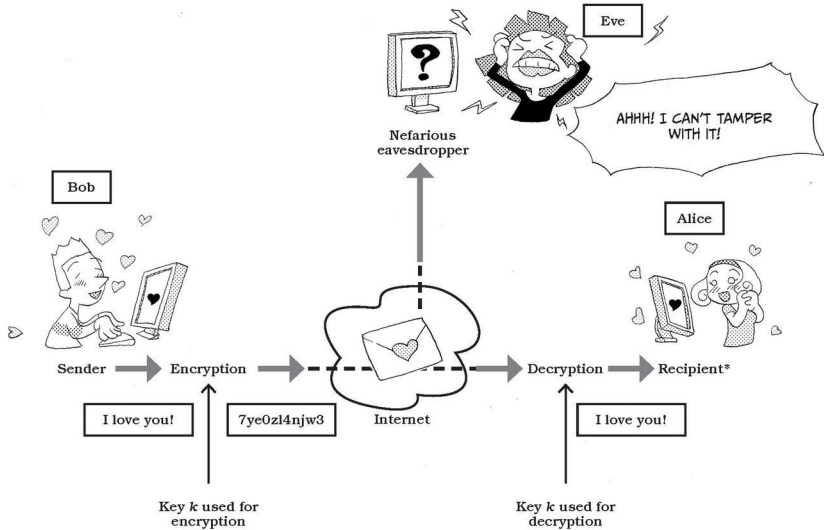
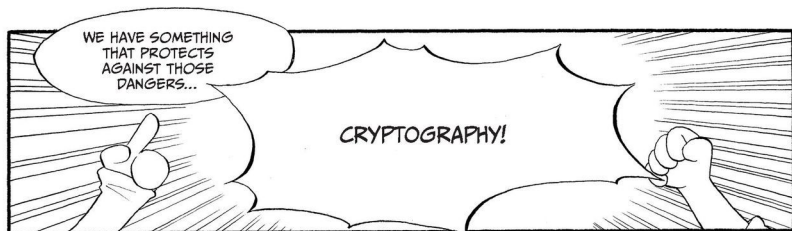
In the computer age, cryptography has become essential to protecting against the falsification, destruction, and interception of information.





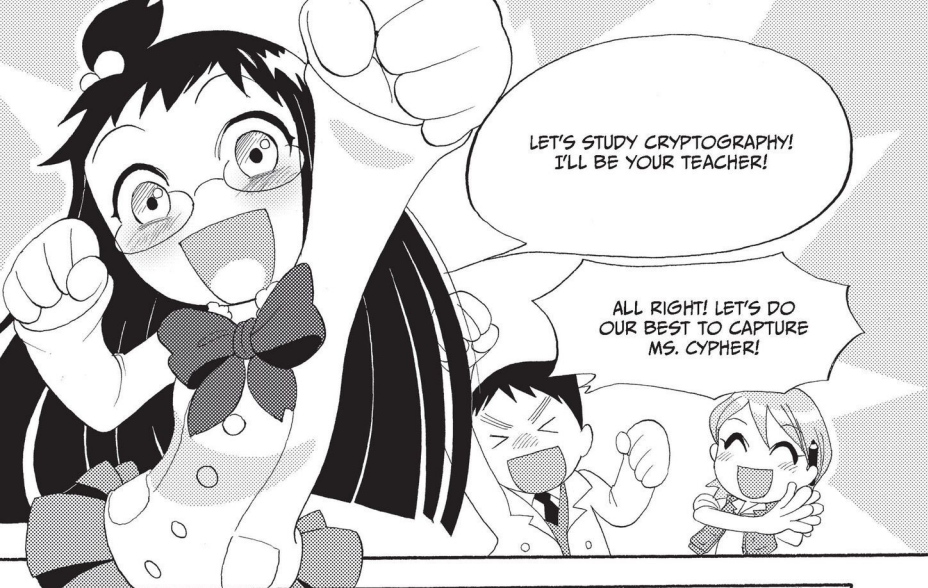
HE HATES ME?!





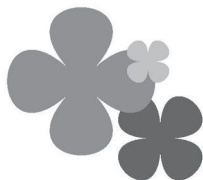
* The recipient doesn't have to be a person. It could instead be a hard disk, computer memory, or some other type of storage media.

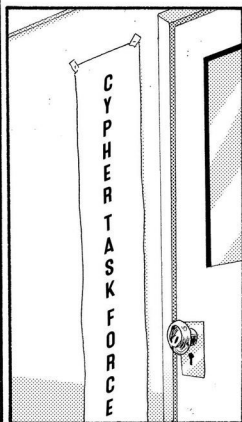




1

THE FOUNDATIONS OF ENCRYPTION





BIG BROTHER, IT'S
TIME TO GET YOU UP
TO SPEED ON THIS
CASE! LISTEN UP!

WAG

WAG

YEAH!

HEH...

SO WHAT'S A
REPORTER LIKE YOU
DOING HERE?

THIS THEFT IS GOING
TO BE A BIG STORY!
I WANT TO COVER IT
THOROUGHLY. ♡

WHAT IS A CIPHER?

A BIG STORY... THE
PRESSURE IS ON!

HMM...

FIRST...

LET'S TALK
ABOUT WHY THIS
PHANTOM THIEF
CALLS HERSELF
"MS. CYPHER".

SHE STEALS OTHER
PEOPLE'S TREASURES, SO
SHE GOES BY THE NAME
CYPHER!*

THAT'S
SIMPLE!

TALK ABOUT
SIMPLE...

cipher

NO, CIPHER! IT'S A WAY
OF WRITING A SECRET
MESSAGE.

* THE WORD FOR TREASURE IS SAIFU IN
JAPANESE, WHICH SOUNDS LIKE CIPHER.

I SEE, A
CIPHER.

MAYDAY = DISTRESS SIGNAL

ROGER = TRANSMISSION RECEIVED

OH!

I KNOW SOME
FAMOUS ONES,
LIKE WHEN A PILOT
SAYS "MAYDAY"
OR "ROGER."



CLOSE BUT NOT
QUITE. THOSE
AREN'T ACTUALLY
CIPHERS.

WAG WAG

WELL,
WHAT ARE
THEY?

IF THEY
AREN'T
CIPHERS...

THESE KINDS OF WORDS ARE
ONLY USED AMONG PEERS IN A
PARTICULAR FIELD. YOU COULD
CALL THESE WORDS JARGON
OR PROCEDURE WORDS.

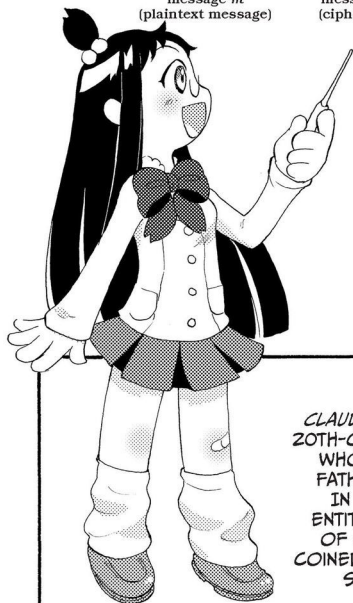
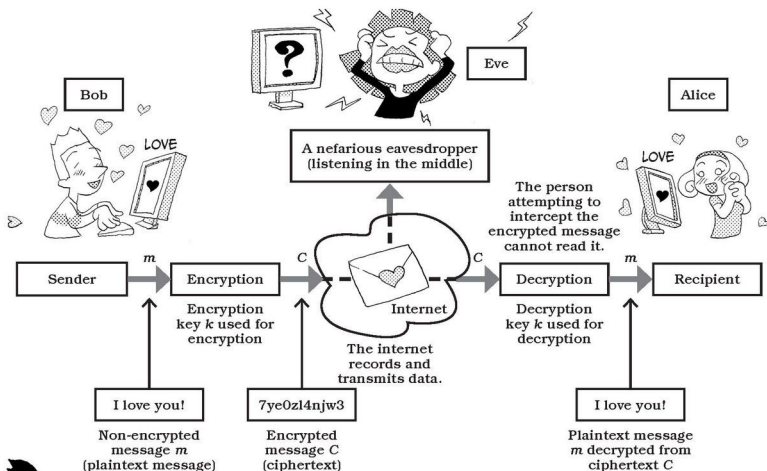
code

THEY'RE ALSO
KNOWN AS
CODES.

BUT WHAT WE'RE
STUDYING...

ARE CIPHERS!

cipher



LET'S LOOK AT HOW CIPHERS
WORK USING SHANNON'S
ENCRYPTION MODEL.

CLAUDE SHANNON (1916-2001) WAS A 20TH-CENTURY ENGLISH MATHEMATICIAN WHO CAME TO BE KNOWN AS THE FATHER OF INFORMATION THEORY. IN 1948, HE WROTE AN ARTICLE ENTITLED "A MATHEMATICAL THEORY OF COMMUNICATION" IN WHICH HE COINED THE TERM **BIT** AND DEVELOPED SEVERAL CONCEPTS KEY TO CRYPTOGRAPHY.



SHANNON'S ENCRYPTION MODEL

Here's what the process looks like, broken down into steps.

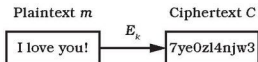
You start with a *plaintext* (m), which is just an unencrypted, normal message.

Plaintext $m =$ I love you!

You end up with *ciphertext* (C), which is the encrypted message.

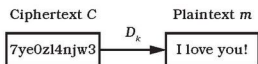
Ciphertext $C =$ 7ye0zl4njw3

Encryption is the process that transforms the plaintext to a ciphertext:

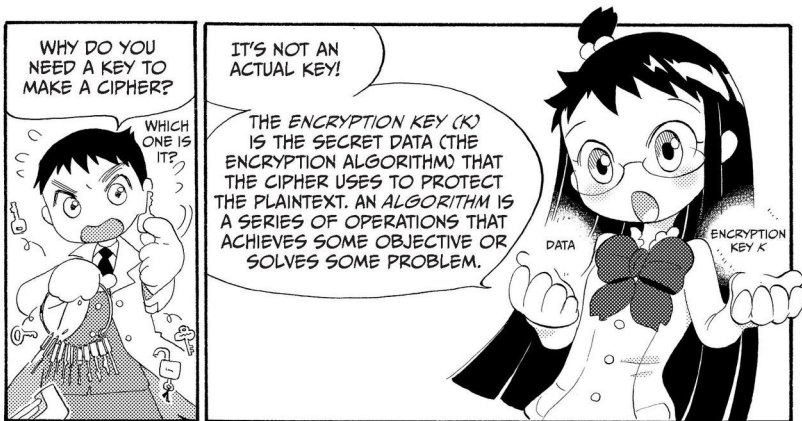


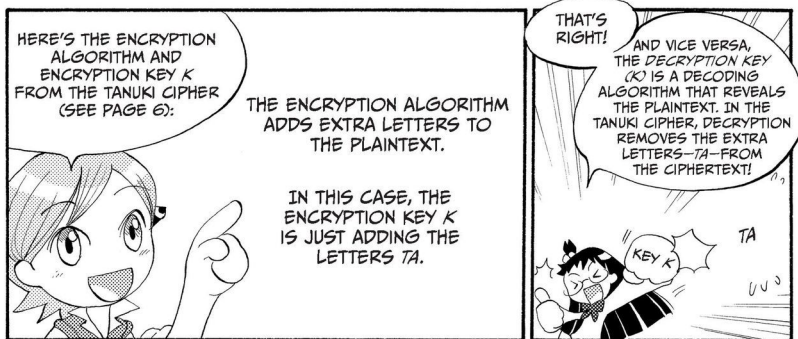
where E_k is encryption using an encryption key k .

Decryption is the process that restores the plaintext from a ciphertext:



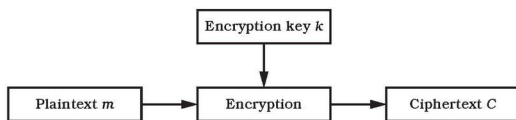
where D_k is decryption using a decryption key k .



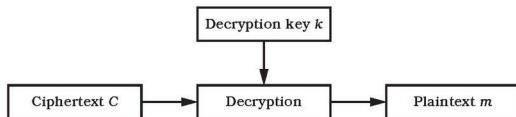


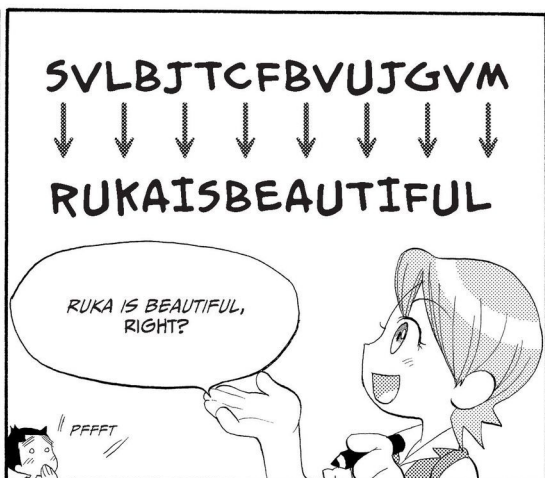
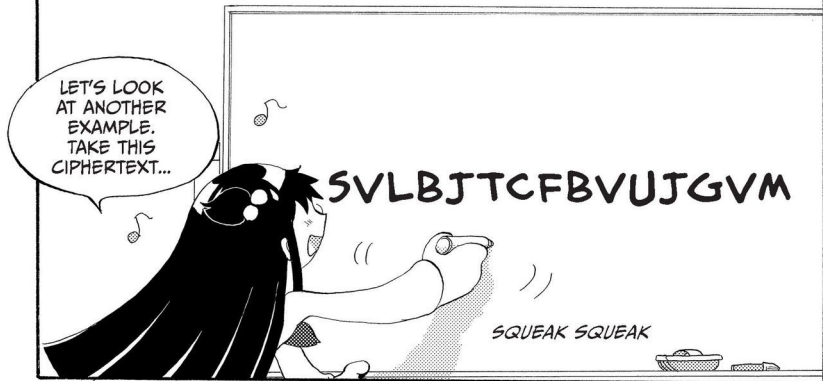
THE RELATIONSHIP BETWEEN THE ENCRYPTION KEY AND THE DECRYPTION KEY

The sender encrypts their message. Using the sender's plaintext message m and encryption key k on the encryption algorithm results in ciphertext C .



The recipient decodes the ciphertext. Ciphertext C is decrypted into plaintext m by using the ciphertext C and decryption key k on the decryption algorithm.





SHE'S
ABSOLUTELY
CORRECT!

DECRYPTION WITH
DECRYPTION KEY K
MOVES EACH LETTER
BACK ONE SPACE TO
ITS ORIGINAL POSITION.

RUKA-SENSEI
IS A LITTLE
SCARY...

BUT CAN'T A CIPHER
LIKE THAT BE
INSTANTLY DECODED?

INFORMATION
SECURITY
HANDBOOK

GEH, THE
WEIGHT OF
KNOWLEDGE...

THAT'S A VERY SIMPLE CIPHER.
CIPHERS HAVE BECOME MUCH
MORE SOPHISTICATED. AFTER
ALL, IF YOU WANT TO PASS
SECRET MESSAGES, YOU HAVE
TO DEAL WITH EAVESDROPPERS
TRYING TO DECIPHER THEM.

LET'S TAKE A LOOK AT
SOME CLASSIC ENCRYPTION
METHODS TO GET A BETTER
UNDERSTANDING OF THIS
GAME OF CAT AND MOUSE!

CLASSIC ENCRYPTION METHODS

Although classic encryption methods are no longer used because they're not secure, they're still useful for learning how encryption works. Let's look at how some of these historical ciphers worked in practice.

CAESAR CIPHER

In the previous example, you saw the Caesar cipher in action. The *Caesar cipher* uses an algorithm that creates the ciphertext by shifting each letter of plaintext n letters. By way of example, let's try to encrypt the title of the Japanese fairytale *Momotaro*. If $n = 3$, we would move each letter forward by three.

... L **M** N O **P** Q R ...

DO THE SAME WITH THE REST OF THE LETTERS:

O → R T → W A → D R → U

Once you've done this, a ciphertext is formed.

M O M O T A R O Plaintext m



P R P R W D U R Ciphertext C

Letters at the end of the alphabet just wrap around to the beginning.

X → A Y → B Z → C

This cipher is named after the Roman soldier and statesman Julius Caesar (Gaius Iulius, 100–44 BCE). Caesar used this cipher during the Gallic Wars to correspond with his allies unbeknownst to his rivals.



* IN JAPANESE, DIE SOUNDS THE SAME AS RHINO.

It's pretty easy to decrypt a plain Caesar cipher, which is a type of *substitution cipher*. Substitution ciphers that replace one letter of the plaintext with another letter to make the ciphertext, as the Caesar cipher does, are called simple substitution ciphers.

Assume that the 26 letters of the English alphabet have been converted as follows:



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

↓ ↓ ↓

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

M O M O T A R O Plaintext m

D G D G Z G K G Ciphertext C

σ? HOW DO YOU SAY THAT?

THAT'S THE
GREEK LETTER
SIGMA.

POLYALPHABETIC CIPHER

A cipher that divides the plaintext into fixed-size blocks of n letters and varies the number of positions each letter shifts within each block is known as a *polyalphabetic cipher*. This makes a substitution cipher even sneakier.

Here we see conversion rule δ (delta) in action for a block of four letters ($n = 4$):

Conversion rule $\square =$

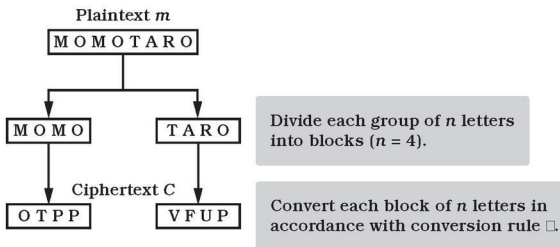
First letter \rightarrow Move by 2 letters

Second letter \rightarrow Move by 5 letters

Third letter \rightarrow Move by 3 letters

Fourth letter \rightarrow Move by 1 letter

The resulting cipher works like this:



The process to decrypt the ciphertext works the same way, just in reverse.



TRANSPOSITION CIPHER

A cipher that divides the plaintext into fixed-size blocks of n letters and then changes the sequence of the letters in each block is called a *transposition cipher*. For example, if $n = 4$, the substitution rule τ (tau) gives us the following:

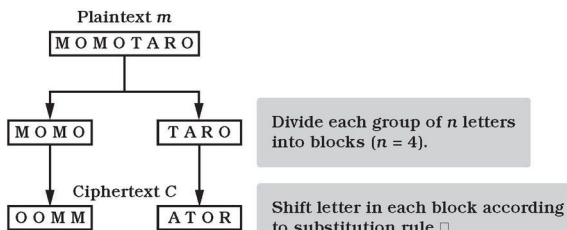
$$\tau = \begin{matrix} & 1 & 2 & 3 & 4 \\ \begin{matrix} \downarrow \\ 3 \\ \downarrow \end{matrix} & 2 & 1 & 3 & 4 \end{matrix}$$

Here's what is going on in this transposition cipher:

Substitution rule $\square =$

- First letter \rightarrow Becomes second letter
- Second letter \rightarrow Becomes fourth letter
- Third letter \rightarrow Becomes first letter
- Fourth letter \rightarrow Becomes third letter

Thus, the following cipher is born:



In this cipher, the rearrangement of the letters makes up the algorithm, and the number of letters per block and the substitution rule τ are the encryption key.



CIPHER SECURITY

ALTHOUGH THE CAESAR CIPHER WAS INVENTED MORE THAN 2,000 YEARS AGO...

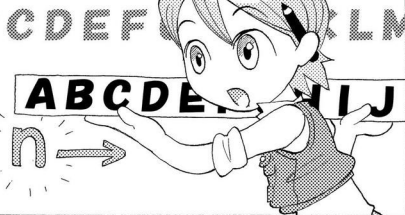


WE USE THE CONCEPTS OF ALGORITHMS AND KEYS...

TO DESCRIBE THE SAME IDEAS IN MODERN CRYPTOGRAPHY.

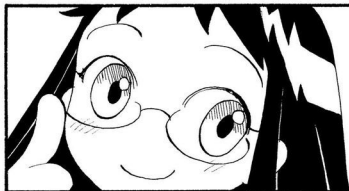
THE CAESAR CIPHER'S ENCRYPTION ALGORITHM...

SHIFTS PLAINTEXT LETTERS BY N NUMBER OF LETTERS.



FOR INSTANCE,

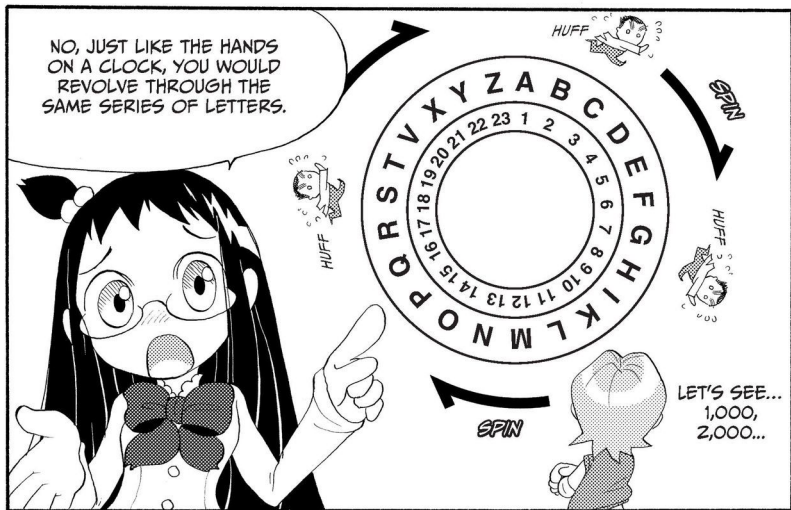
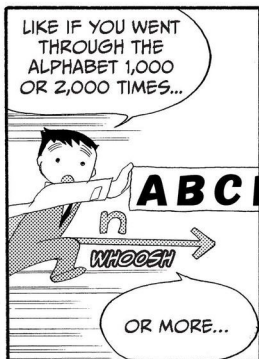
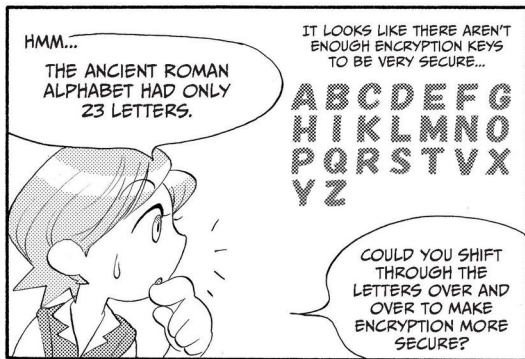
$N = 3$ IS THE ENCRYPTION KEY THAT CAESAR USED.



SVL
↓ ↓

THE PROBLEM I SHOWED YOU EARLIER—SVLBJTFCFBVJJGVM—IS AN EXAMPLE OF ENCRYPTING WITH THE CAESAR CIPHER.

RUKAISBEAUTIFUL



SO, IF AN EAVESDROPPER
NOTICED THE STRUCTURE
OF A CAESAR CIPHER THAT
USED THE ANCIENT ROMAN
ALPHABET...

THEY COULD FIGURE
OUT THE KEY AND
DECODE THE CIPHER
IN FEWER THAN 23
ATTEMPTS.



SO IT'S BETTER TO
USE JAPANESE THAN IT
IS TO USE THE ROMAN
ALPHABET!

AHA!

NICE
IDEA!

BECAUSE WE HAVE
HIRAGANA, KATAKANA,
AND KANJI, THERE
ARE MORE THAN
10,000 KEYS.

TO PREVENT YOUR
CIPHERTEXT FROM
BEING DECODED...

IT'S BEST TO HAVE A
VERY LARGE NUMBER
OF KEYS.



LET'S TAKE A LOOK
AT HOW MANY
KEYS THERE ARE IN
OTHER CIPHERS.



THE SUBSTITUTION CIPHER'S KEY SPACE

Let's think more about how many potential keys there might be for substitution ciphers and how easy such a cipher would be to break. You saw that a Caesar cipher using the ancient Roman alphabet had only 22 possible keys (a *key space* of 22), so breaking it wouldn't be too hard. The key space of the Caesar cipher is limited because it depends on shifting the alphabet to determine the plaintext letter's ciphertext substitution. In Caesar's version of the Caesar cipher, the substitution letters are simply the alphabet shifted by three positions, but you could take a different approach by scrambling the positions of the letters entirely.

Instead of shifting the letters to determine letter substitutions, you could substitute each letter of the alphabet with any other letter as long as each plaintext letter corresponds with one unique substitution. Your plaintext letters would be the alphabet in order:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Then one example of a substitution with a scrambled alphabet could be written out like this:

QWERTYUIOPASDFGHJKLZXCVBNM

Each letter of the alphabet would be substituted with a corresponding letter in the scrambled version of the alphabet. You can simply substitute according to the position of the letters in the alphabet, so *A* would be substituted as *Q*, *B* as *W*, and so on. The scrambled alphabet would be the cipher's key.

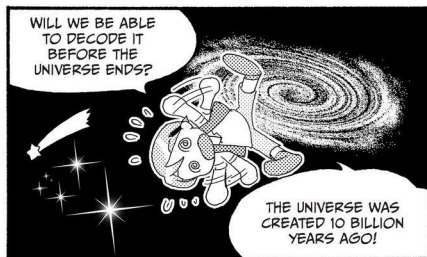
If we used our modern alphabet with 26 letters, we would have a large number of ways to potentially arrange those letters for a key:

$${}_{26}P_{26} = 26! = 26 \times 25 \times 24 \times \dots \times 3 \times 2 \times 1 \approx 4.03291461 \times 10^{26}$$

The letter *P* is used to represent a *permutation*, which is a unique arrangement of letters in a sequence. The subscript number before the *P* means that there are 26 total characters, and the subscript number after the *P* means we are using all 26 of them. The exclamation mark means *factorial*, or the product of the integer and all the positive integers less than it. This formula makes sense if you think about it this way:

- The first letter could be assigned any of the 26 letters, so there are 26 to choose from.
- The second letter could be any of the remaining 25 letters, so there are 25 to choose from. For two letters, there are 26×25 possibilities.
- The third letter could be any of the remaining 24 letters.
And so on until . . .
- The last letter will be whichever letter remains—there is one option for the last letter.

Given $4.03291461 \times 10^{26}$ possible permutations of letters, if a computer were asked to generate all these arrangements until it found the correct key to break the cipher, then, even if it were to perform an exhaustive search at a rate of 100,000,000 keys per second, it could take a preposterous period of up to 128,000,000,000 years.



PERMUTATION, COMBINATION, AND FREQUENCY ANALYSIS

Let's look at permutations in more detail. A *permutation*, expressed as P , is a method of selecting r things from n possibilities and putting them into unique arrangements. By unique arrangements, we mean that, for instance, ABCDE is not the same as EDCBA. Those are considered two different arrangements.

If we selected 5 letters out of the 26 letters of the alphabet, r would be 5 and n would be 26. Then we would use this formula to find the number of arrangements of 5 letters selected from 26:

$${}_nP_r = n \times (n-1) \times (n-2) \times \dots \times (n-r+1) = \frac{n!}{(n-r)!}$$

Thus,

$${}_{26}P_5 = \frac{26!}{(26-5)!} = \frac{26!}{21!}$$

Because all the factors in $21!$ are in $26!$, you can cancel out the numbers up to and including 21:

$$\frac{26!}{21!} = 26 \times 25 \times 24 \times 23 \times 22 = 7,893,000$$

In contrast, if r things are selected from n possibilities and order does not matter, this is called *combination* and is expressed as C in the combination formula:

$${}_nC_r = \frac{{}_nP_r}{r!} = \frac{n!}{(n-r)!r!}$$

If we're still selecting 5 letters out of 26 possible letters, we would get this:

$$\frac{26!}{(21! \cdot 5!)}$$

The factors would cancel like in the permutation example, which would leave you with this:

$$\frac{(26 \cdot 25 \cdot 24 \cdot 23 \cdot 22)}{(1 \cdot 2 \cdot 3 \cdot 4 \cdot 5)} = 65,780$$

Because order does not matter, the arrangements ABCDE and EDCBA would be considered the same, meaning that there are fewer possible arrangements when using combinations rather than permutations.

Searching for the correct key within the large space of $4.03291461 \times 10^{26}$ permutations is theoretically possible, but in practical terms, it's a *computationally infeasible problem*. That said, these types of ciphers are susceptible to a cryptographic attack known as *frequency analysis*, which assumes that the frequency of letters that appear in the plaintext and the frequency of letters that appear in the ciphertext are consistent.

For example, the letter *E* is the most common letter in English. That means if *Z* is the most common letter in the encrypted text, you can reasonably assume that the corresponding letter in the plaintext is *E*. Frequency analysis attacks are one of the most effective attacks on simple substitution ciphers.

THE POLYALPHABETIC CIPHER'S KEY SPACE

Let's assume that a given block is n characters long. Because we don't know how many spaces the first letter has been shifted, we make 26 attempts to figure it out. Each letter shift in the block is independent from the other shifts, so the second letter could also take 26 attempts, same with the third letter, and so on, until we arrive at the n th letter.

Using what we just learned about permutations, combinations, and frequency analysis, r would be block size n , and the possible correct values for n would be the 26 letters of the alphabet and would apply to each letter. Because of this, the total number of keys is as follows:

$$26 \cdot 26 \cdot \dots \cdot 26 \cdot 26 = 26^n$$

If $n = 4$, then the key space would be the following:

$$26^4 = 26 \times 26 \times 26 \times 26 = 456,976$$

As n grows larger, the number of keys rapidly increases. When $n = 10$, the number of possible keys surpasses 140 trillion.



THE TRANSPOSITION CIPHER'S KEY SPACE

The total number of keys for a block of n characters is expressed as follows:

$${}_nP_n = n(n-1)(n-2)\dots \times 3 \times 2 \times 1 = n!$$

So when there are four letters in one block ($n = 4$), the total number of keys k is as follows:

$$4! = 4 \times 3 \times 2 \times 1 = 24$$

As with other ciphers, the key space increases as n becomes larger, and the security of the encryption becomes stronger. Note that $n = 26$ would give the same key space as the substitution cipher.

THEN COULD A
SUBSTITUTION CIPHER
WITH A LARGE ENOUGH
KEY SPACE BE SECURE?
IT IS SIMPLE...

~~~~~  
HMMMMM?

SUBSTITUTION

POLYALPHABETIC

THE POLYALPHABETIC  
CIPHER SEEMS REALLY  
COMPLICATED.

SIMPLE SUBSTITUTION  
CIPHERS CAN BE EASILY  
CRACKED USING BASIC  
DECODING CLUES.

THE GOLD BUG

Edgar Allen Poe

GOLD BUG? IS THAT  
A PICTURE BOOK OF  
INSECTS?

I'M RICH!

IT'S A FAMOUS MYSTERY  
STORY THAT DEALS WITH  
CRYPTOGRAPHY.

OOOOOH,  
A STORY!

HERE'S PART OF "THE  
GOLD BUG" CIPHER:

53+++305))6\*:4826)  
4\*.J4\*);806\*:48+  
8¶60))85;1†(;:;\*8+83(88)

SIMPLE SUBSTITUTION CIPHERS  
CAN BE DECODED USING  
FREQUENCY ANALYSIS!

THAT'S HOW THE  
PROTAGONIST IN "THE  
GOLD BUG" SOLVES  
THE CIPHER.

e

the

WITHOUT A  
DOUBT!

BECAUSE WE KNOW THAT  
THE MOST COMMON  
LETTER IN ENGLISH IS E  
AND THE MOST COMMON  
WORD IS THE...

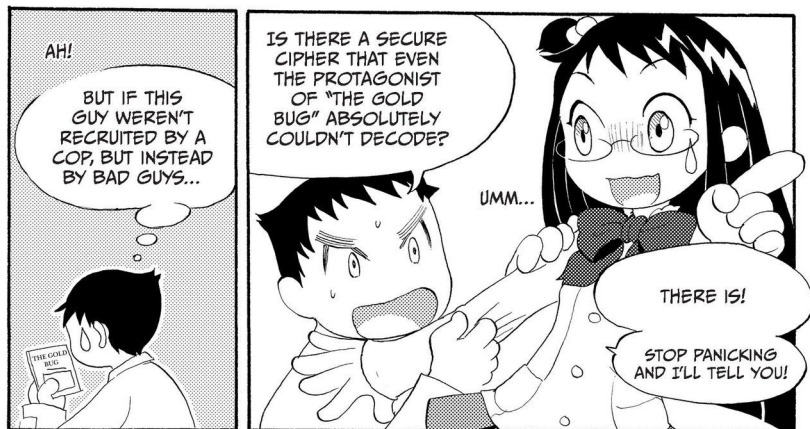
WE CAN START BY  
ASSUMING THAT THE  
MOST FREQUENTLY USED  
SYMBOL, 8, IS E AND THAT  
THE MOST FREQUENTLY  
USED COMBINATION OF  
SYMBOLS, ;AB, IS THE.

BRILLIANT!

I WISH I COULD  
RECRUIT HIM TO THE  
FORCE!

WHEN A CIPHERTEXT  
IS LONG, THESE KINDS  
OF CLUES INCREASE  
IN NUMBER, MAKING  
DECODING EASIER.

BUT WHEN YOU HAVE  
A SHORT SENTENCE,  
DECODING IS  
TOUGH!



## WHEN BREAKING CLASSICAL ENCRYPTION IS POSSIBLE

Typically, breaking classical encryption is possible under these conditions:

1. When you understand the encryption algorithm
2. When there is data about the statistical properties of the encrypted plaintext; for example, the frequency with which a letter or word appears
3. When you have a large number of encrypted example sentences

### PERFECTLY SECURE ENCRYPTION

By generating a random number to use as a one-time pad—a key that is as long as the plaintext, is used once, and is then discarded—you can produce a cipher that is computationally secure. More specifically, you would be generating a ciphertext by applying a string of random numbers to plaintext  $m$  that is the same length as the plaintext. This ciphertext cannot be decrypted without knowledge of the key. Unfortunately, a one-time pad is not practical because it is difficult to distribute the one-time pad to all the communicating parties ahead of time.

Gilbert Vernam devised a cipher that uses a one-time pad, the Vernam cipher, in 1917 and patented it a couple of years later. During World War II, Claude Shannon (see page 19) established that this cipher is unbreakable and in 1949 publicly published his mathematical proof.

Here is a simple example of a Vernam cipher. You start by converting the alphabet to numerical values in a process known as *character encoding*:

|    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|
| A  | B  | C  | D  | E  | F  | G  | H  | I  | J  | K  | L  | M  |
| 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 |
| N  | O  | P  | Q  | R  | S  | T  | U  | V  | W  | X  | Y  | Z  |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

- Convert each letter to a numerical value (e.g., A becomes 0).

Plaintext

|    |    |    |    |    |   |    |    |
|----|----|----|----|----|---|----|----|
| M  | O  | M  | O  | T  | A | R  | O  |
| ↓  | ↓  | ↓  | ↓  | ↓  | ↓ | ↓  | ↓  |
| 12 | 14 | 12 | 14 | 19 | 0 | 17 | 14 |

- Add each number in the sequence to the corresponding one-time use number.

Random number  
sequence  
(encryption key)

|    |    |    |    |    |   |    |    |
|----|----|----|----|----|---|----|----|
| 12 | 14 | 12 | 14 | 19 | 0 | 17 | 14 |
| +  | +  | +  | +  | +  | + | +  | +  |
| 9  | 20 | 15 | 23 | 27 | 2 | 15 | 8  |
| =  | =  | =  | =  | =  | = | =  | =  |
| 21 | 34 | 27 | 37 | 46 | 2 | 32 | 22 |

- Calculate the remainder when each number is divided by 26.

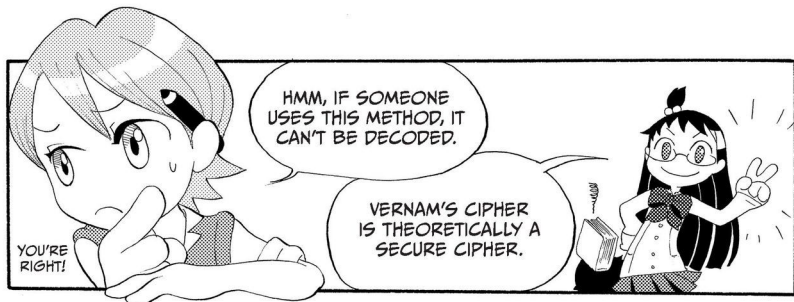
|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 21 | 34 | 27 | 37 | 46 | 2  | 32 | 22 |
| ÷  | ÷  | ÷  | ÷  | ÷  | ÷  | ÷  | ÷  |
| 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 |
| =  | =  | =  | =  | =  | =  | =  | =  |
| 21 | 8  | 1  | 11 | 20 | 2  | 6  | 22 |

- Convert the numbers to letters using the character encoding.

Ciphertext

|    |   |   |    |    |   |   |    |
|----|---|---|----|----|---|---|----|
| 21 | 8 | 1 | 11 | 20 | 2 | 6 | 22 |
| ↓  | ↓ | ↓ | ↓  | ↓  | ↓ | ↓ | ↓  |
| V  | I | B | L  | U  | C | G | W  |





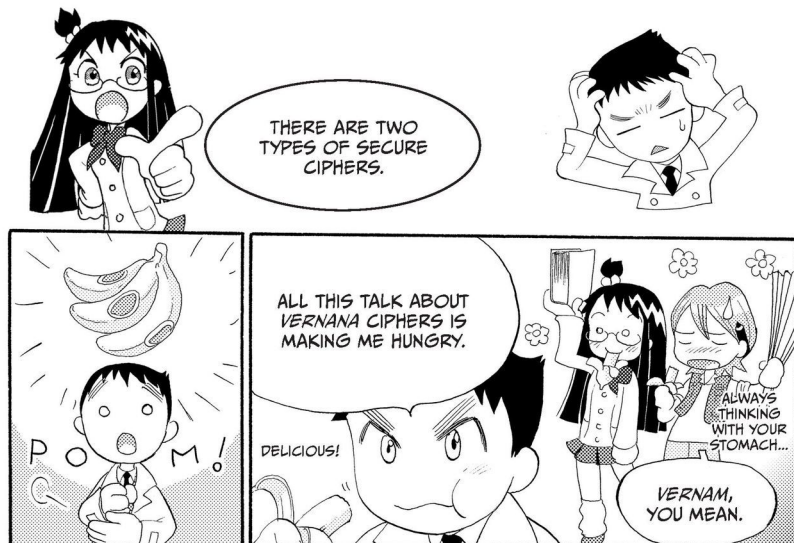
## SECURE CIPHERS

### 1. Fail-safe ciphers

A fail-safe cipher like Vernam's cipher is theoretically unbreakable.

### 2. Computationally secure ciphers

Although computationally secure ciphers can be decoded, doing so is computationally infeasible because it's so time and labor intensive that it isn't worth trying. These ciphers are what we use today.





EVENING FALLS

THANKS FOR  
CHOOSING  
USAGI'S DINER!

DINNERTIME!

THANK YOU FOR  
WAITING.

UNTIL NEXT  
TIME—

Usagi's  
Diner

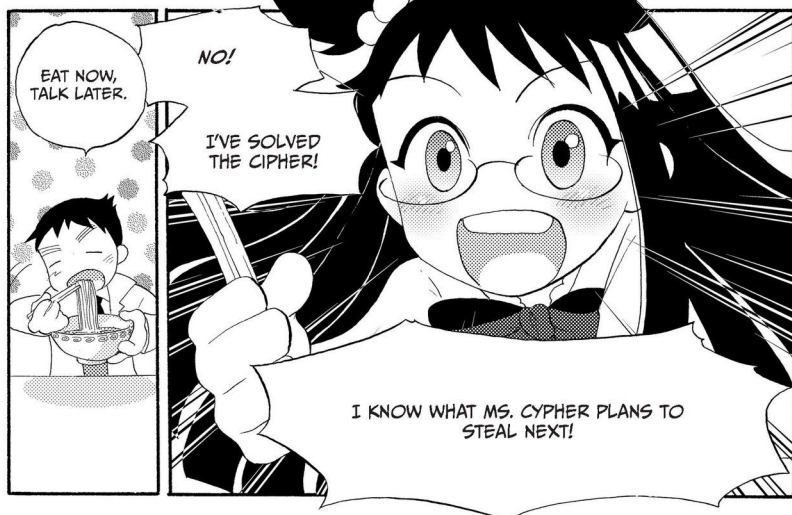
DELICIOUS!

SLURP  
SLUUUUURP

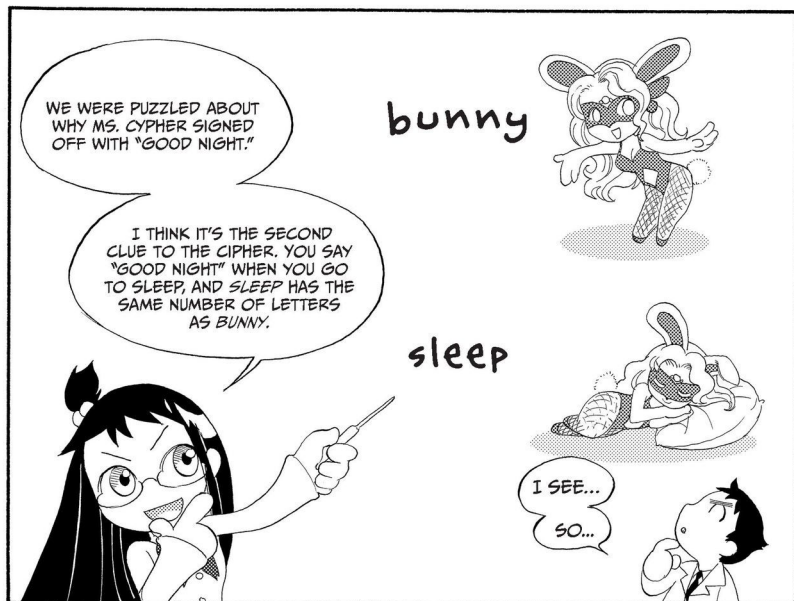
RAMEN...  
USAGI\*...  
MS. CYPHER...

GOOD NIGHT...

\* USAGI MEANS RABBIT OR BUNNY IN JAPANESE.







IT'S A CAESAR CIPHER!  
IF YOU SHIFT EACH  
LETTER IN *BUNNY*  
FORWARD BY  
17 LETTERS...

bunny  
↓ ↓ ↓ ↓ ↓ 17 LETTERS  
sleep

YOU GET *SLEEP*!

MOVING THE LETTERS  
17 PLACES IS THE KEY!

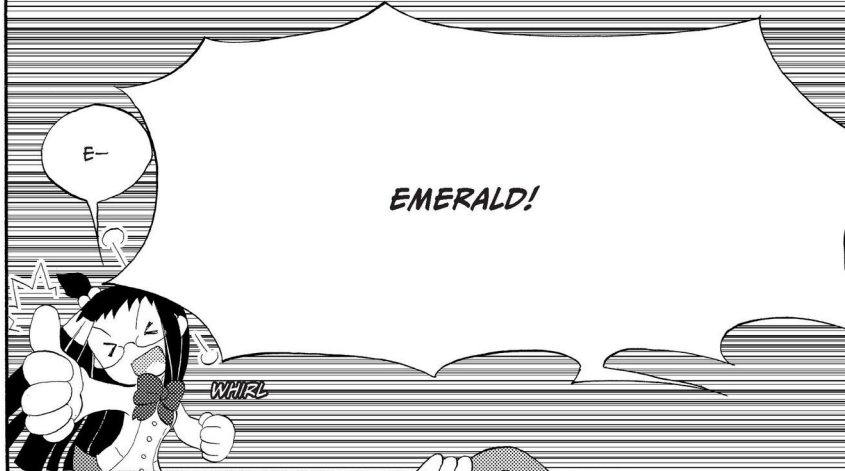
YES!

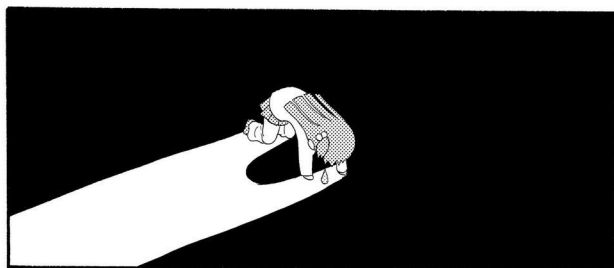
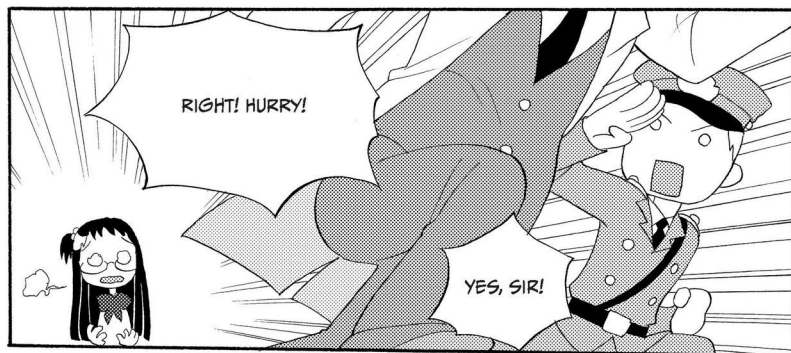
SO IF WE MOVE EACH  
LETTER IN *VDVRCU*  
17 SPACES...

THE SOLUTION IS...

gulp

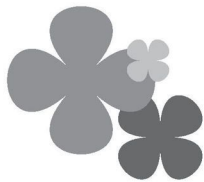






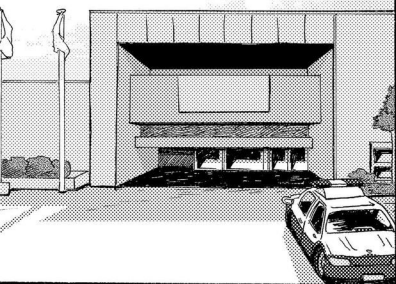
# 2

## SYMMETRIC-KEY ALGORITHMS



## BINARY DIGITS AND LOGICAL OPERATORS

MARBLE ART  
MUSEUM

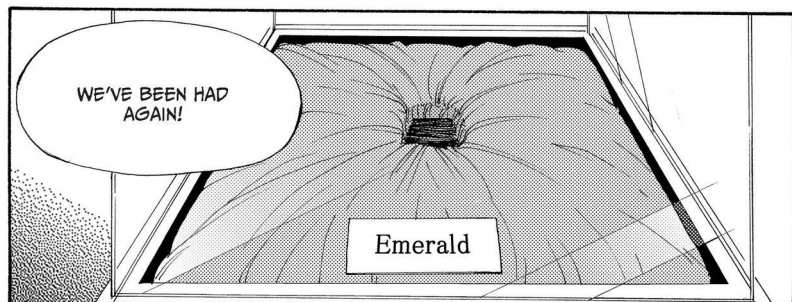


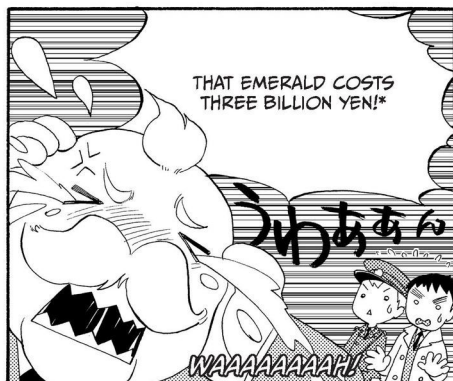
## INTERNATIONAL TREASURES EXHIBITION



WE'VE BEEN HAD  
AGAIN!

Emerald





THAT EMERALD COSTS  
THREE BILLION YEN!\*

うああん

WAAAAAAAAAH!

\* ALMOST 30 MILLION DOLLARS



WAS THE  
EXHIBITION CASE  
LOCKED?

OF COURSE!  
BUT—



IT MUST HAVE BEEN OPENED  
AT SOME POINT. BECAUSE  
YOUR PRECIOUS GEM WAS  
REELED RIGHT UP FROM  
THE CEILING.



HI!



YOU AGAIN!

HOW DO YOU  
KNOW THE THIEF'S  
MODUS OPERANDI?

FWIP

THIS WAS  
DELIVERED TO THE  
NEWSROOM!





CHECK OUT MY CATCH OF THE DAY:  
YOUR PRECIOUS EMERALD ♥

-MS. CYPHER

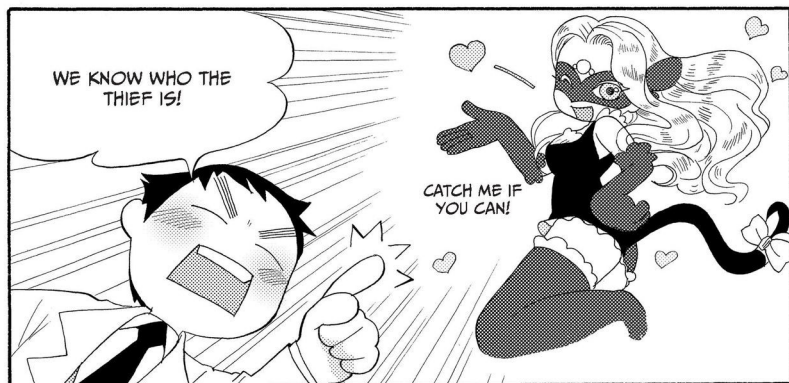
PS: IF YOU WANT TO  
MEET ME IN THE FLESH,  
THEN FIGURE THIS OUT!

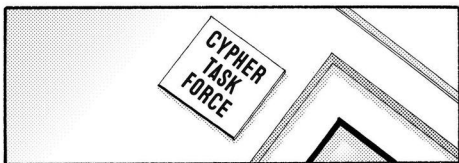


00110001 00101011 00110001 00111101 00110000









00110001 00101011 00110001 00111101 00110000



YOU GOT IT! COMPUTERS  
STORE DATA AS 1s AND 0s,  
OR BINARY DIGITS.

A SINGLE BINARY DIGIT  
(A 1 OR 0) IS CALLED  
A BIT.

We usually use the base-10 decimal number system, but the binary system is in base 2, which uses only two numbers to count.

When you count past 1 in binary, you run out of digits and must carry the 1 to the next largest place value. For example, the number 2 is 10 in binary.

A group of eight bits in a row is called a *byte*. Each bit represents a 0 or 1, so a byte has a total of  $2^8 = 256$  possible combinations. Each combination represents a different piece of information.

| Binary number | Decimal number | Hexadecimal number | Binary number | Decimal number | Hexadecimal number |
|---------------|----------------|--------------------|---------------|----------------|--------------------|
| 0000          | 0              | 0                  | 1000          | 8              | 8                  |
| 0001          | 1              | 1                  | 1001          | 9              | 9                  |
| 0010          | 2              | 2                  | 1010          | 10             | A                  |
| 0011          | 3              | 3                  | 1011          | 11             | B                  |
| 0100          | 4              | 4                  | 1100          | 12             | C                  |
| 0101          | 5              | 5                  | 1101          | 13             | D                  |
| 0110          | 6              | 6                  | 1110          | 14             | E                  |
| 0111          | 7              | 7                  | 1111          | 15             | F                  |

As numbers become larger, the number of digits required to represent them in binary grows rapidly, so *hexadecimal notation* is often used instead. Hexadecimal notation is in base 16, using digits 0 through 9 and letters A through F, as shown in this table. This means that after you count up to 9, you use A instead of 10, B instead of 11, and so on up to F. The prefix 0x is added to hexadecimal numbers, so for example, the decimal number 10 is 0xA in hexadecimal.

0 IS 0,  
1 IS 1,  
2 IS 10...

UNLIKE HISTORICAL CIPHERS,  
WHICH OPERATE ON LETTERS,  
MODERN ENCRYPTION METHODS  
OPERATE ON BINARY NUMBERS.



THE ROW AND COLUMN HEADINGS ARE GIVEN IN HEXADECIMAL NOTATION. AT THE TOP IS THE HEXADECIMAL VALUE OF THE FOUR LOWER-ORDER BITS, AND ON THE LEFT IS THE HEXADECIMAL VALUE OF THE FOUR HIGHER-ORDER BITS.

|                   |   | Lower-order bits |     |       |   |   |   |   |     |  |  |  |  |  |  |  |  |
|-------------------|---|------------------|-----|-------|---|---|---|---|-----|--|--|--|--|--|--|--|--|
| Higher-order bits |   | 0                | 1   | 2     | 3 | 4 | 5 | 6 | 7   |  |  |  |  |  |  |  |  |
|                   | 0 | NUL              | DLE | space | 0 | @ | P | ` | p   |  |  |  |  |  |  |  |  |
|                   | 1 | SOH              | DC1 | !     | 1 | A | Q | a | q   |  |  |  |  |  |  |  |  |
|                   | 2 | STX              | DC2 | "     | 2 | B | R | b | r   |  |  |  |  |  |  |  |  |
|                   | 3 | ETX              | DC3 | #     | 3 | C | S | c | s   |  |  |  |  |  |  |  |  |
|                   | 4 | EOT              | DC4 | \$    | 4 | D | T | d | t   |  |  |  |  |  |  |  |  |
|                   | 5 | ENQ              | NAK | %     | 5 | E | U | e | u   |  |  |  |  |  |  |  |  |
|                   | 6 | ACK              | SYN | &     | 6 | F | V | f | v   |  |  |  |  |  |  |  |  |
|                   | 7 | BEL              | ETB | '     | 7 | G | W | g | w   |  |  |  |  |  |  |  |  |
|                   | 8 | BS               | CAN | (     | 8 | H | X | h | x   |  |  |  |  |  |  |  |  |
|                   | 9 | HT               | EM  | )     | 9 | I | Y | i | y   |  |  |  |  |  |  |  |  |
|                   | A | LF               | SUB | *     | : | J | Z | j | z   |  |  |  |  |  |  |  |  |
|                   | B | VT               | ESC | +     | ; | K | [ | k | {   |  |  |  |  |  |  |  |  |
|                   | C | FF               | FS  | ,     | < | L | \ | l |     |  |  |  |  |  |  |  |  |
|                   | D | CR               | GS  | -     | = | M | ] | m | }   |  |  |  |  |  |  |  |  |
|                   | E | SO               | RS  | .     | > | N | ^ | n | ~   |  |  |  |  |  |  |  |  |
|                   | F | SI               | US  | /     | ? | O | _ | o | del |  |  |  |  |  |  |  |  |

SO IF WE CONVERT THE REST OF THE CODE MS. CYPHER LEFT, WE GET...

| Binary number | Hexadecimal number | ASCII |
|---------------|--------------------|-------|
| 00110001      | 31                 | 1     |
| 00101011      | 2B                 | +     |
| 00110001      | 31                 | 1     |
| 00111101      | 3D                 | =     |
| 00110000      | 30                 | 0     |

↓

$1 + 1 = 0$

BUT DOESN'T  $1 + 1 = 2$ ?

IS MS. CYPHER AN IDIOT?

OF COURSE SHE'S NOT!

THIS IS AN XOR OPERATION...

ALSO CALLED AN EXCLUSIVE OR OPERATION.

IT'S A LOGICAL OPERATION USED IN CRYPTOGRAPHY!

WAAA!

XO\* JAR?

SORE?

SOAR?

THIS IS ALL WAY OVER MY HEAD, SO I'LL JUST GO GET A SNACK.

NOT SO FAST!

\* XO SAUCE IS A CHINESE CONDIMENT.



MS. CYPHER HAS USED A BITWISE OPERATION, WHICH IS A CALCULATION THAT WORKS ONLY ON BINARY NUMBERS (1s AND 0s). ALL COMPUTER CALCULATIONS ARE BITWISE OPERATIONS!



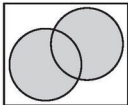
HERE'S A FIGURE TO ILLUSTRATE.  $A$  AND  $B$  ARE INPUTS FOR THE BITWISE OPERATIONS, AND FOR EACH OPERATION, WE GET AN OUTPUT THAT IS A BITWISE NUMBER (EITHER 1 OR 0).

AND operations are sometimes referred to as conjunction logic, and OR operations are sometimes referred to as disjunction logic. XOR operations are a combination of conjunction and disjunction logic.

OR operation

$A + B$

| A | B | $A + B$ |
|---|---|---------|
| 0 | 0 | 0       |
| 1 | 0 | 1       |
| 0 | 1 | 1       |
| 1 | 1 | 1       |

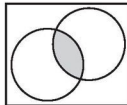


If either  $A$  or  $B$  is a 1, the output is 1.

AND operation

$A \cdot B$

| A | B | $A \cdot B$ |
|---|---|-------------|
| 0 | 0 | 0           |
| 1 | 0 | 0           |
| 0 | 1 | 0           |
| 1 | 1 | 1           |

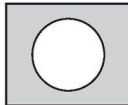


If both  $A$  and  $B$  are 1, the output is 1.

NOT operation

$\bar{A}$

| A | $\bar{A}$ |
|---|-----------|
| 1 | 0         |
| 0 | 1         |

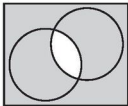


The output is the opposite of  $A$ . If  $A$  is 1, the output is 0; if  $A$  is 0, the output is 1.

NAND operation

$\overline{A \cdot B}$

| A | B | $\overline{A \cdot B}$ |
|---|---|------------------------|
| 0 | 0 | 1                      |
| 1 | 0 | 1                      |
| 0 | 1 | 1                      |
| 1 | 1 | 0                      |

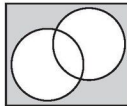


If either  $A$  or  $B$  is 0, the output is 1; otherwise it is 0.

NOR operation

$\overline{A + B}$

| A | B | $\overline{A + B}$ |
|---|---|--------------------|
| 0 | 0 | 1                  |
| 1 | 0 | 0                  |
| 0 | 1 | 0                  |
| 1 | 1 | 0                  |

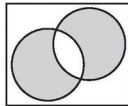


If both  $A$  and  $B$  are 0, the output is 1; otherwise it is 0.

XOR operation

$A \oplus B = \overline{A \cdot B} + \overline{A \cdot B}$

| A | B | $A \oplus B$ |
|---|---|--------------|
| 0 | 0 | 0            |
| 1 | 0 | 1            |
| 0 | 1 | 1            |
| 1 | 1 | 0            |



If  $A$  and  $B$  are different, the output is 1; otherwise it is 0.

SO...

WHEN YOU USE THE XOR OPERATION ON TWO DIFFERENT VALUES THE OUTPUT IS 1. IF THE INPUT VALUES ARE THE SAME, THE OUTPUT IS 0.

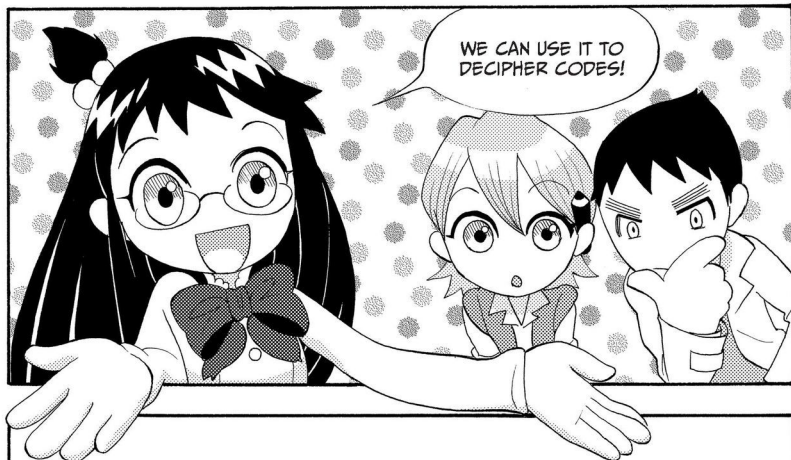
THE SYMBOL FOR AN XOR IS  $\oplus$  AND IS USED LIKE THIS:  
 $1 \oplus 0 = 1, 1 \oplus 1 = 0.$



OKAY, BUT HOW IS THIS USEFUL?







As an example, let's perform an XOR operation using the plaintext 1101 and the encryption key 1001.

$$\begin{array}{rcl}
 1101 & \square & 1001 & = & 0100 \\
 \text{Plaintext} & & \text{Encryption key} & & \text{Ciphertext}
 \end{array}$$

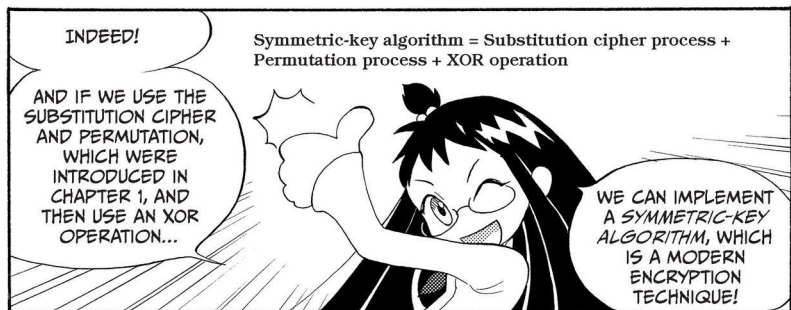
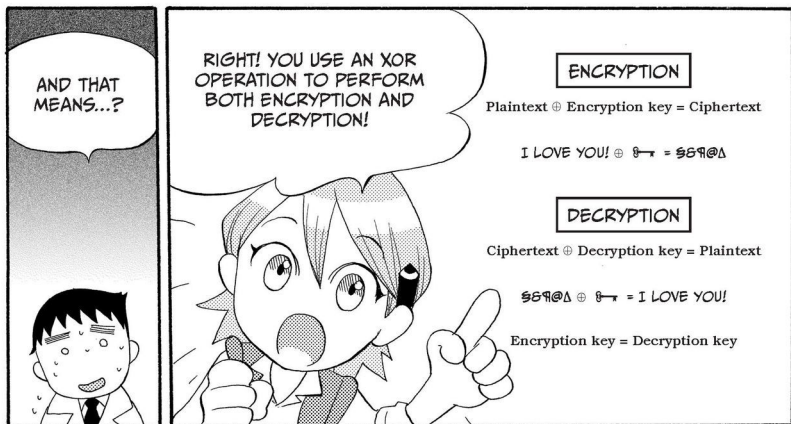
The result of the operation is the ciphertext 0100. Next, we'll perform an XOR operation using the ciphertext 0100 and the decryption key 1001.

$$\begin{array}{rcl}
 0100 & \square & 1001 & = & 1101 \\
 \text{Ciphertext} & & \text{Decryption key} & & \text{Plaintext}
 \end{array}$$

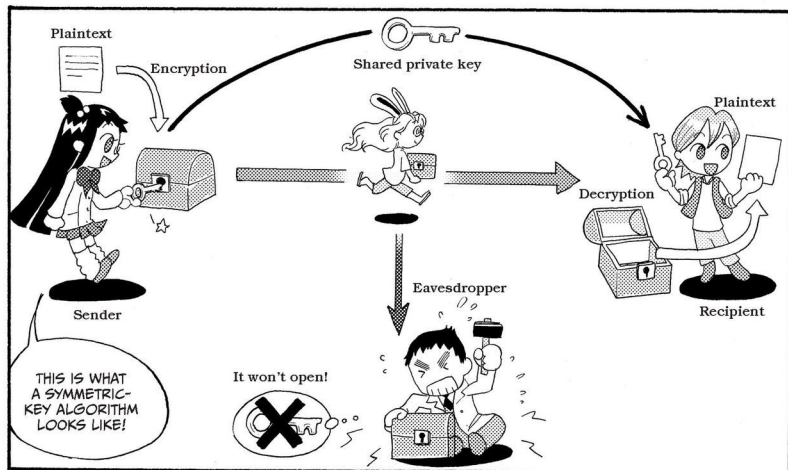
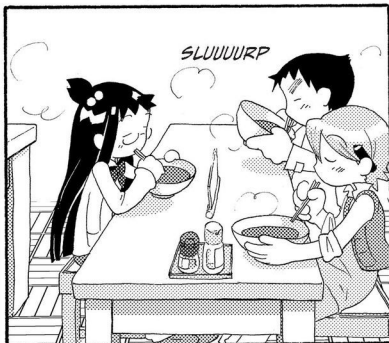
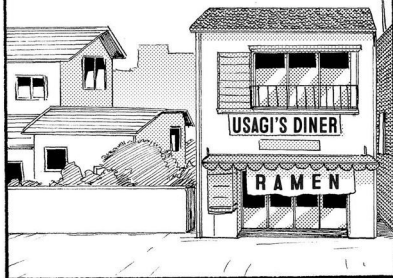
The result of the operation is the plaintext 1101. When you perform the XOR operation using the ciphertext 0100 and the plaintext 1101, you get the encryption/decryption key:

$$\begin{array}{rcl}
 0100 & \square & 1101 & = & 1001 \\
 \text{Ciphertext} & & \text{Plaintext} & & \text{Decryption key} = \text{Encryption key}
 \end{array}$$

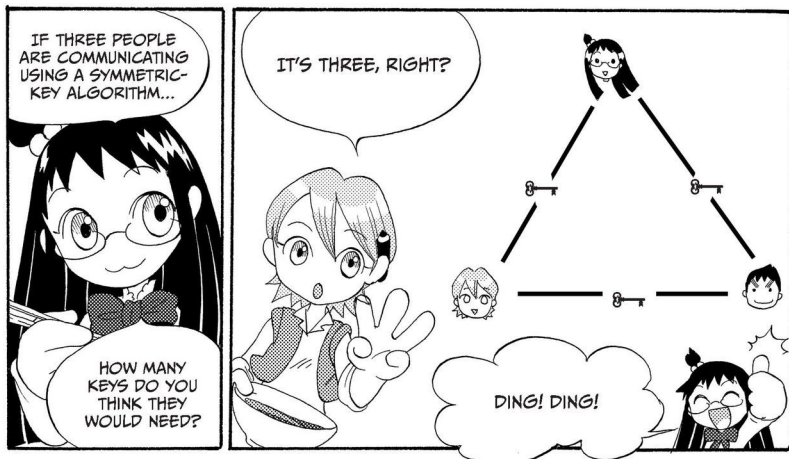
This means that as long as we have two of the three pieces of data—the plaintext, encryption/decryption key, or ciphertext—we can derive the remaining piece of data.

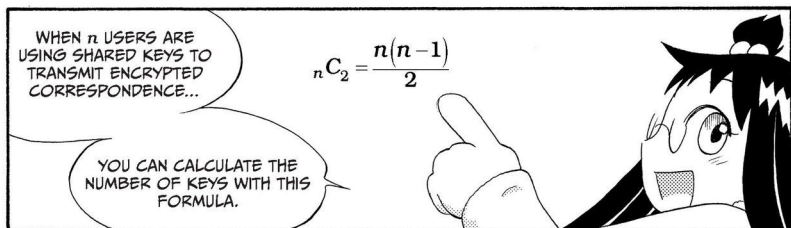
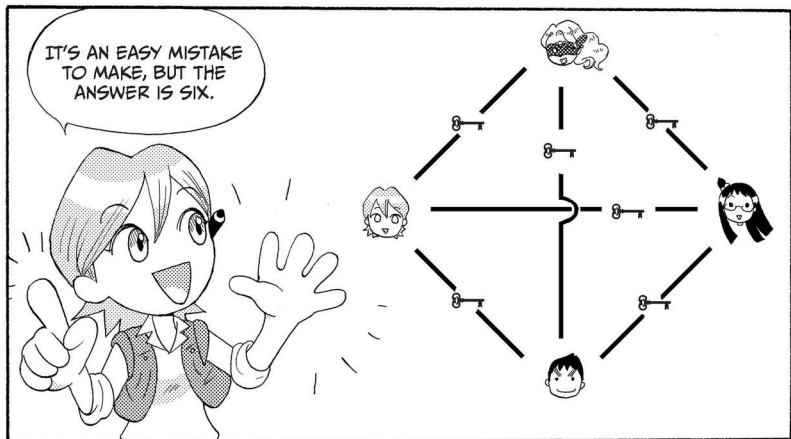
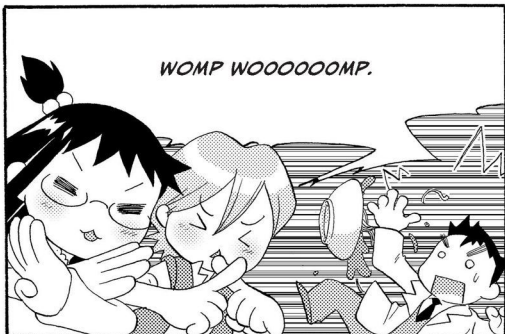


## SYMMETRIC-KEY ALGORITHMS

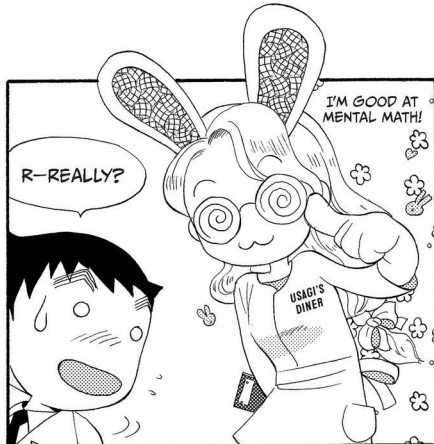
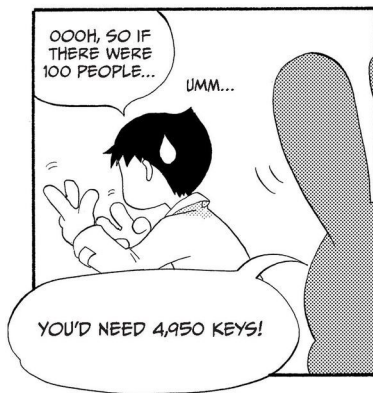


BECAUSE BOTH PARTIES SHARE THE SAME KEY, SYMMETRIC-KEY CRYPTOGRAPHY IS ALSO KNOWN AS SHARED-KEY CRYPTOGRAPHY OR SECRET-KEY CRYPTOGRAPHY. MOST TRADITIONAL ENCRYPTION SCHEMES USE A SHARED KEY.

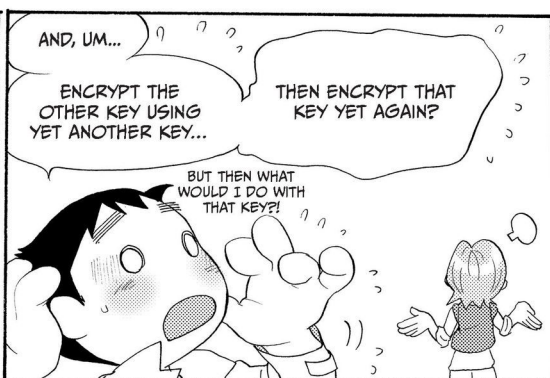


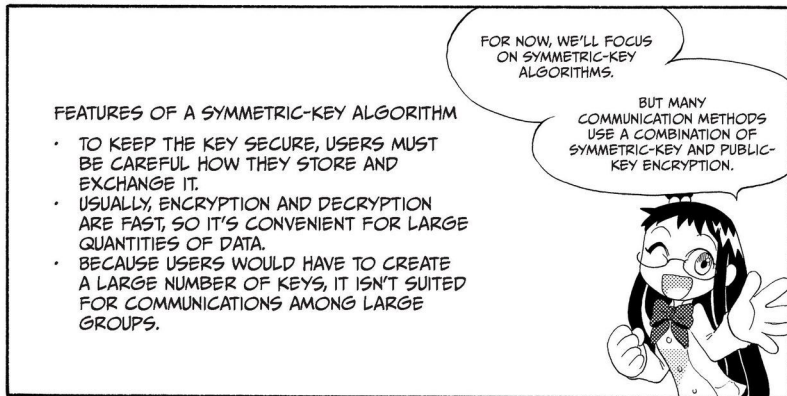
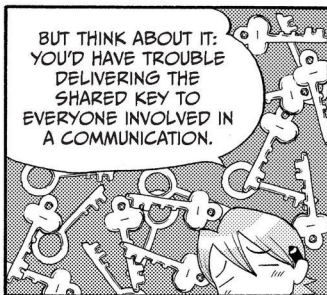












\* PUBLIC-KEY ENCRYPTION USES SEPARATE BUT MATHEMATICALLY PAIRED KEYS FOR ENCRYPTION AND DECRYPTION. THIS IS DIFFERENT FROM SYMMETRIC-KEY ENCRYPTION, WHERE THE SAME KEY IS USED TO BOTH ENCRYPT AND DECRYPT A MESSAGE.

THERE ARE TWO MAIN TYPES OF SYMMETRIC-KEY ALGORITHMS:

STREAM CIPHERS

BLOCK CIPHERS

WHAT'S THE DIFFERENCE?

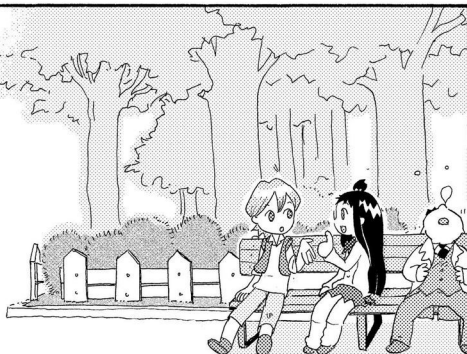
A STREAM CIPHER ENCODES EACH BIT OR BYTE INDIVIDUALLY IN SUCCESSION.

I'LL EXPLAIN THAT IN A MOMENT.

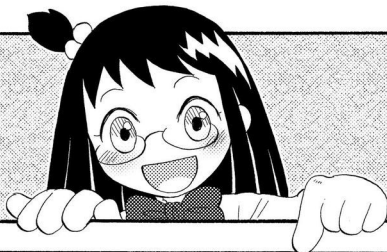
A BLOCK CIPHER SPLITS PLAINTEXT AND CRYPTOGRAPHIC DATA INTO SECTIONS OF UNIFORM LENGTHS (BLOCKS). THEN IT PERFORMS ENCRYPTION AND DECRYPTION ONE BLOCK AT A TIME.

FIRST, LET'S LOOK AT STREAM CIPHERS.

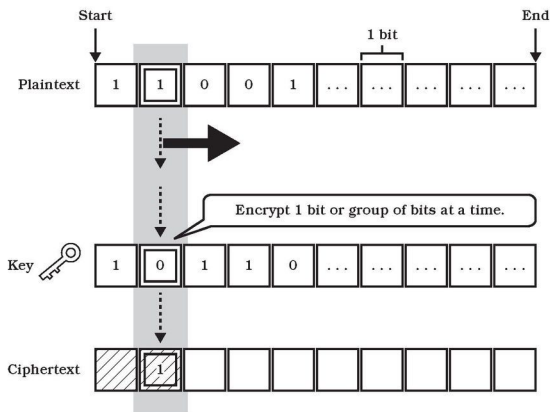
## STREAM CIPHERS



LET'S SEE HOW  
STREAM CIPHERS  
PERFORM  
ENCRYPTION.



In a stream cipher, the key is a long, pseudorandom sequence of numbers. You can encrypt or decrypt with the key by performing XOR operations on each bit or byte data, as shown in the following figure. Stream ciphers are generally less computationally intensive than block ciphers. Some examples of stream ciphers include RC4 and SEAL.





WHAT'S A PSEUDORANDOM  
NUMBER SEQUENCE?

IT'S A SERIES OF  
NUMBERS THAT APPEAR  
COMPLETELY RANDOM.

CAN'T YOU JUST MAKE  
A TRULY RANDOM  
SEQUENCE?

IF YOU MADE THE KEY A STRING  
OF RANDOM NUMBERS THAT'S  
AS LONG AS OR LONGER THAN  
THE PLAINTEXT, YOU COULD  
USE A ONE-TIME PAD TO  
HAVE A SECURE CIPHER.

YES, BUT IT'S IMPRACTICAL TO  
CREATE A KEY THAT LARGE AND  
TO GENERATE A TRULY RANDOM  
NUMBER SEQUENCE.

INCIDENTALLY, BOTH STREAM  
AND BLOCK CIPHERS ARE  
VULNERABLE TO *BRUTE-  
FORCE* ATTACKS, IN WHICH AN  
ATTACKER CAN UNCOVER THE  
CORRECT KEY BY TRYING ALL  
POSSIBLE KEYS.

THIS MEANS THAT  
SECURITY IS GUARANTEED  
ONLY UP TO A POINT,  
DEPENDING ON HOW MUCH  
COMPUTATIONAL POWER  
AN ATTACKER HAS.

SIGH

HEY! WHERE DID  
BIG BROTHER GO?

NOW THAT YOU  
MENTION IT...

WHEE!



## BLOCK CIPHERS

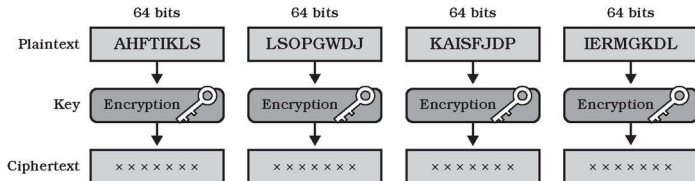
CYBER  
PASH  
FORCE

NOW LET'S LEARN  
ABOUT THE STRUCTURE  
OF BLOCK CIPHERS.

HOW DO THEY  
DIFFER FROM  
STREAM CIPHERS?

A BLOCK CIPHER  
IS A METHOD  
OF ENCRYPTING  
BLOCKS OF DATA  
IN SUCCESSION.

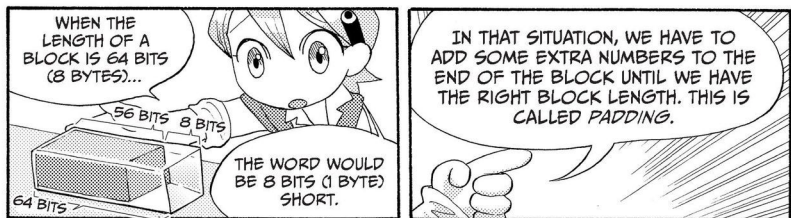
Unlike a stream cipher, which encrypts each bit in succession, a block cipher splits data into blocks of uniform length and then encrypts each block, as shown here.



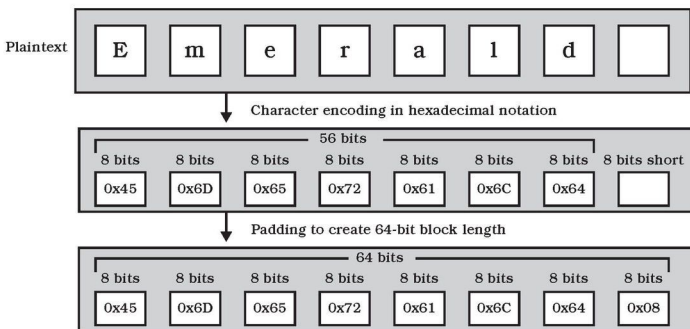
The length of a given block depends on the structure of the cipher used. Block sizes of 64, 128, or 256 bits are the most common for ciphers. Because 1 byte is composed of 8 contiguous bits, it would be reasonable to use 64-bit blocks because you could use 8 characters to satisfy the block size. Using 64-bit blocks allows for faster calculations on a computer.

| Name of cipher | Block length (bits) | Key length (bits) |
|----------------|---------------------|-------------------|
| DES            | 64                  | 64                |
| AES            | 128                 | 128 192 256       |



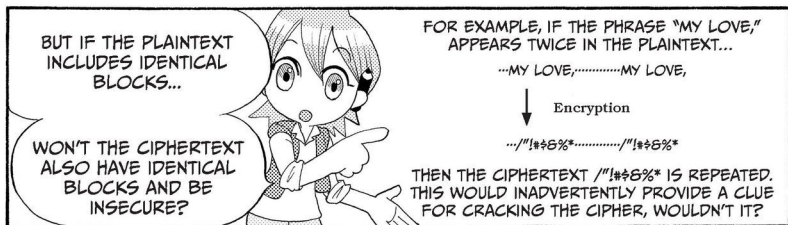
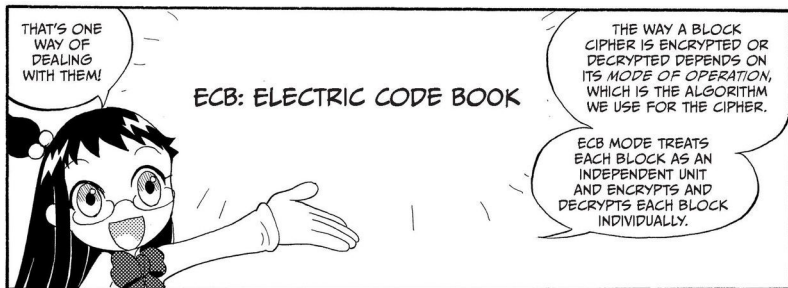


Here's how you would create a block with a length of 64 bits.



Remember: digits that have the prefix 0x are numbers in hexadecimal notation.

The value 0x08 is added to the end of the block so that the block length is 64 bits (8 bytes). The padding is 0x08 because that is the amount of padding that should be removed when the message is decrypted. This is just one way to pad blocks.



## CBC MODE

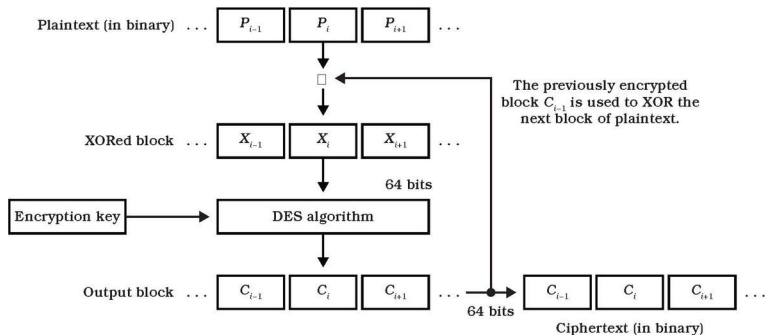
CBC mode outputs different ciphertexts even in cases where the plaintext is the same.

Once the plaintext is cut into blocks, the first block is XORed with an *initialization vector* (an additional cipher block, which is a value known to everyone), and then it's encrypted using a block cipher with a key. The next plaintext block is XORed with the most recently encrypted block and then also encrypted using the block cipher with a key.

This process is done on all of the blocks, making it practically impossible for any connection to be made between blocks. In the following figure, the previous ciphertext block  $C_{i-1}$  and the next plaintext block  $P_i$  are XORed to produce  $X_i$ . In other words,  $X_i$  is calculated by using the following formula:

$$X_i = C_{i-1} \oplus P_i$$

With this method, even if there are blocks of identical plaintext data, there are no identical blocks in the ciphertext.



CBC and ECB are two of the most well-known modes of operation. ECB is no longer used because it is insecure. Other common modes are OFB (output feedback) and CFB (cipher feedback).



## DES CIPHERS



THE FIRST ENCRYPTION STANDARD THAT WAS WIDELY ADOPTED FOR BUSINESS PURPOSES WAS DES...

BUT IT WAS ACTUALLY BASED ON ANOTHER CRYPTOSYSTEM!

The first standardized civilian encryption system was DES (Data Encryption standard). It was based on an earlier system, the Lucifer cipher, which was developed in the early 1970s by Horst Feistel at IBM.

COOL!

CAN YOU GIVE ME AN EXAMPLE OF HOW DES WORKS?

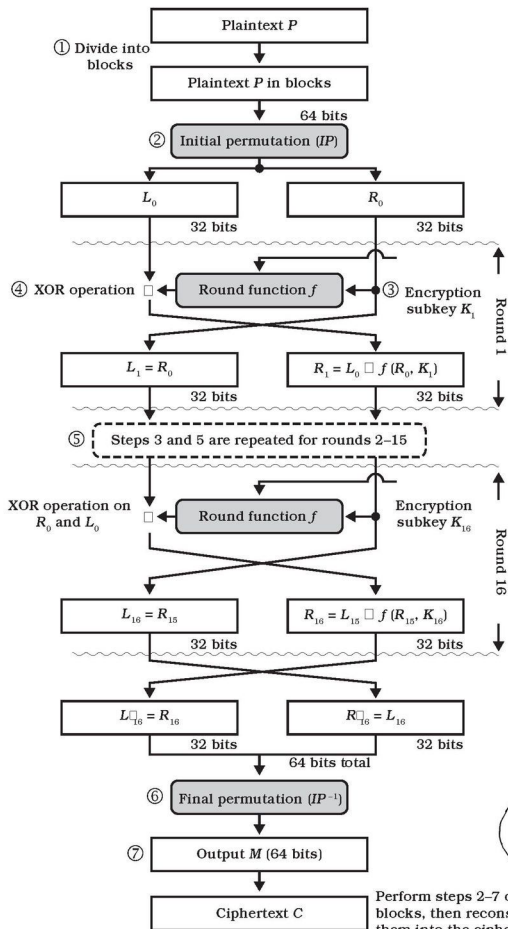
SURE!

LET'S START BY TAKING A LOOK AT FEISTAL CIPHERS, WHICH DES WAS BASED ON. THE CIPHERS YOU'VE SEEN SO FAR HAVE BEEN SIMPLE...

WITH JUST A FEW STEPS PERFORMED ON EACH ENCRYPTED BLOCK. IN PRACTICE, CIPHERS PERFORM A LONGER SERIES OF STEPS.

## THE BASIC CONFIGURATION OF A FEISTEL CIPHER

Each set of steps is called a *round*, and rounds are performed multiple times on a block. The series of steps performed each round is the *round function*. The Feistel cipher uses multiple rounds and subkeys that are derived from one initial key. The following figure outlines Feistel's method of encryption; we'll discuss the details of each round and how the subkeys are created later.



① Divide the plaintext into blocks of 64 bits.

② Perform an initial permutation ( $IP$ ) using an algorithm for each 64-bit block that substitutes and rearranges the bits so that even bits are in the left half of the block and odd bits are in the right half. The block is split in half so that the 32 even bits now on the right side are  $R_0$ , and the left 32 odd bits are  $L_0$ .

③ Use the round function  $f$  with the encryption key ( $K_1$ ). We'll go into more detail about  $f$  later. For these examples, you just need to know that  $f$  is an algorithm that rearranges and substitutes bits in a block.

④ Perform an XOR operation on  $R_0$  and  $L_0$  to create a new right 32-bit block ( $R_1$ ) to replace the original right block. Take the original  $R_0$  and make it into the 32-bit block on the left ( $L_1$ ).

⑤ After completing one round of steps 3 and 4, repeat these steps for rounds 2 through 15.

⑥ Reconstruct the left 32-bit block ( $L_{16}$ ) and the right 32-bit block ( $R_{16}$ ) into one 64-bit block.

⑦ Finally, perform a reversal of the initial permutation ( $IP^{-1}$ ), and one block of encryption is complete.

THE STRUCTURE OF DES IS SIMILAR TO THIS!



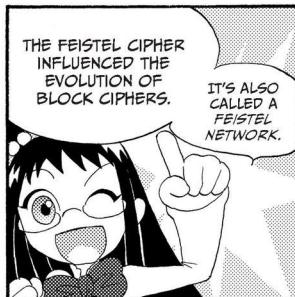
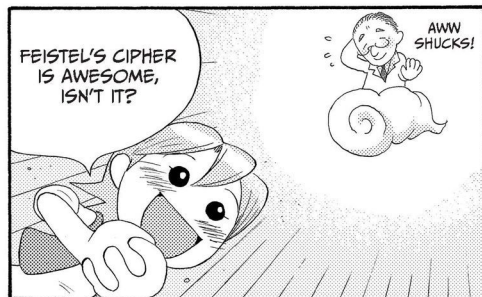


## INVOLUTION

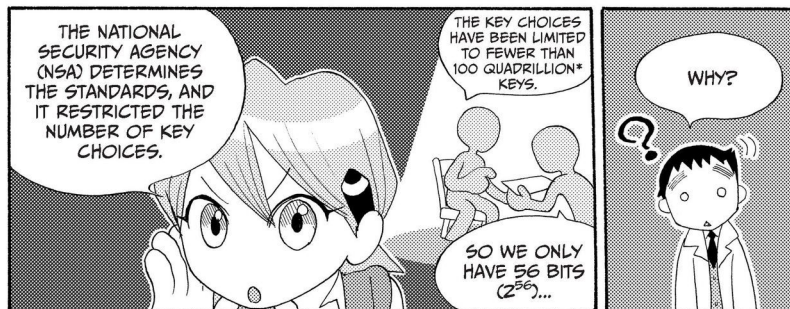
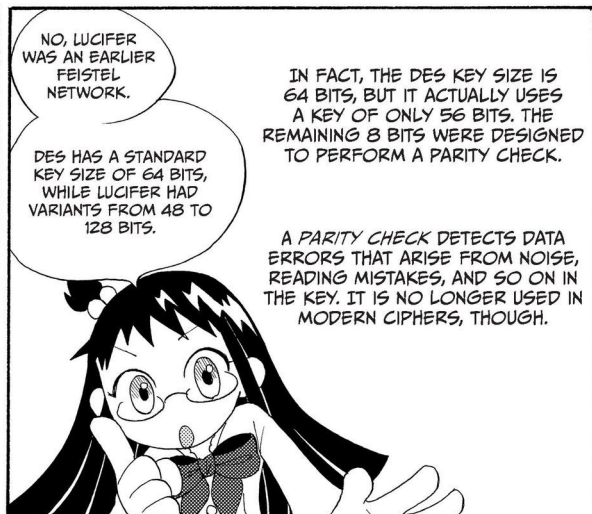
An involution is a function that is its own inverse, which means that if you apply the function twice, you will return the input to its original state. For example, consider a function in which 1 becomes 4, 2 becomes 3, 3 becomes 2, and 4 becomes 1. The values in the list after the conversion are the same as the values before.

| First conversion | Second conversion |
|------------------|-------------------|
| 1 → 4            | 4 → 1             |
| 2 → 3            | 3 → 2             |
| 3 → 2            | 2 → 3             |
| 4 → 1            | 1 → 4             |

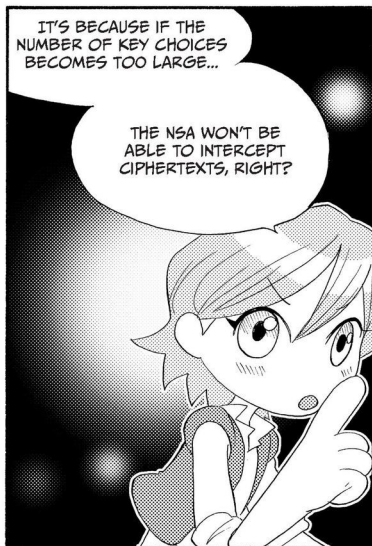
By applying the function twice, we return to all of the original values in the same order.





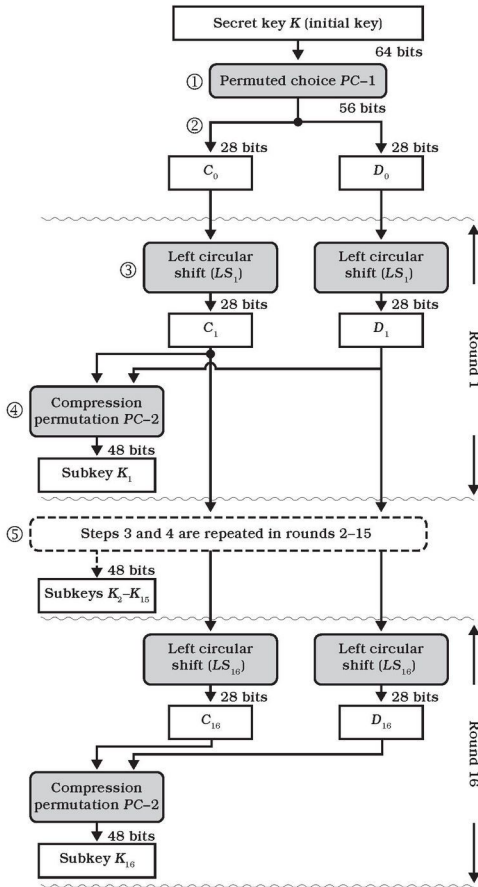


\* ONE HUNDRED MILLION =  $10^8$ , ONE TRILLION =  $10^{12}$ , TEN QUADRILLION =  $10^{16}$ , ONE HUNDRED QUINTILLION =  $10^{20}$



## GENERATING DES ENCRYPTION SUBKEYS

In DES, a different key is used for each round of encryption, so 16 subkeys— $K_1, K_2, K_3, \dots, K_{16}$ —need to be created from the encryption key for each of the 16 rounds. This is the method used to derive subkeys:



① A permutation called permuted choice 1 (PC-1) is performed on the 56 nonparity bits of the initial key. Permuted choice 1 is a special permutation method that transposes each of the 56 bits into specific positions. We won't cover the details of it here.

② The 56 bits are divided into a 28-bit left block ( $C_0$ ) and a 28-bit right block ( $D_0$ ).

③ The bits in  $C_0$  and  $D_0$  are left-circular shifted to create  $C_1$  and  $D_1$ . A left-circular shift moves every bit a certain number of places to the left, and the bits at the beginning of the block are shifted to the very end, as though the block's beginning and end were connected as in a circle.

④  $C_1$  and  $D_1$  are joined. Then subkey  $K_1$  is made through another permutation, permuted choice 2 (PC-2). PC-2 is applied to all 56 bits of  $C_1$  and  $D_1$ , except for the 8 parity bits, resulting in a compressed 48-bit subkey.

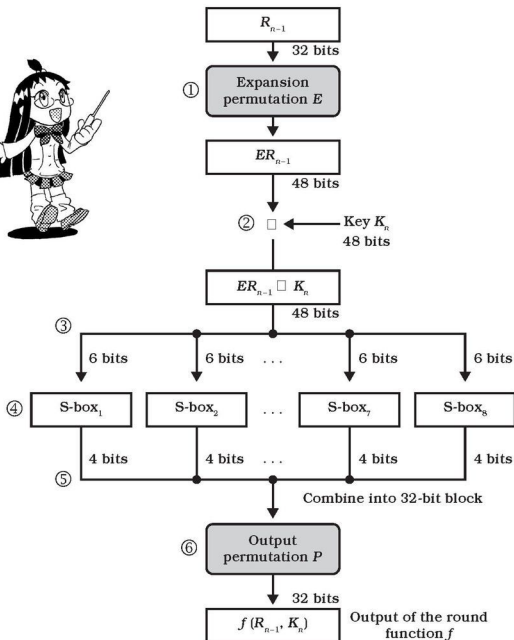
⑤ Steps 3 and 4 are repeated for each round to generate each subkey  $K_i$ . Each of these subkeys will then be used for each round in DES.

To generate the decryption subkeys, the bits are right-circular shifted instead. When the initial key is used for decryption, the subkeys are obtained in reverse order from  $K_{16}$  to  $K_1$ .



## THE DES ROUND FUNCTION F

In a round function, you use an *S-box (substitution box)* to perform substitutions on an input to make a new output. Each S-box in this diagram contains a different permutation for creating the substitutions.



① The DES function only works on 48-bit blocks, so the rightmost block of data, which is only 32 bits, needs to be expanded using a 48-bit expansion permutation  $E$ . This results in the output  $ER_{n-1}$ , which is a 48-bit block.

② Perform an XOR operation on the data and the subkey.

③ Separate the result of the operation into eight sets of 6-bit blocks each.

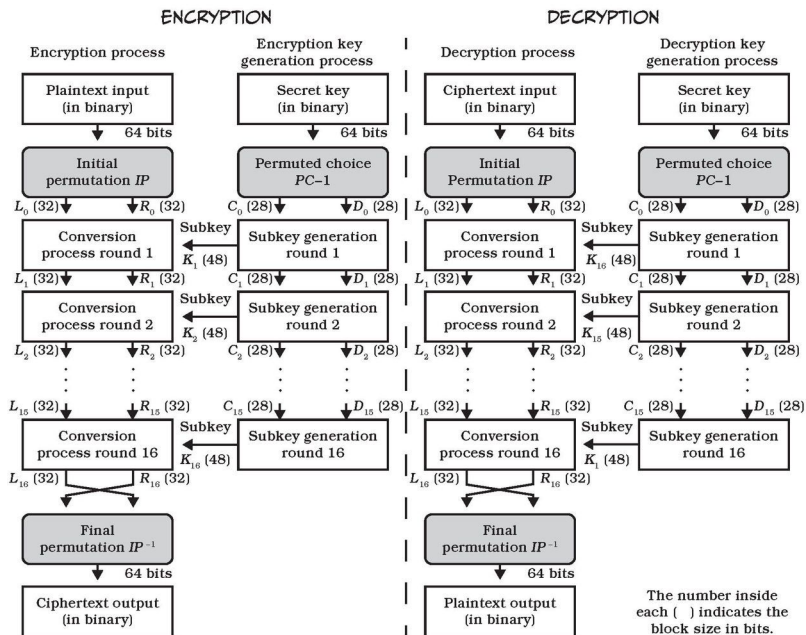
④ Substitute each 6-bit set of data with 4 bits using S-boxes 1 through 8.

⑤ Combine the S-box output data sequentially to produce a 32-bit block.

⑥ Finally, apply permutation  $P$  to the data to yield the output of function  $f$ .

## THE FULL STRUCTURE OF DES ENCRYPTION AND DECRYPTION

The full DES encryption and decryption processes with all the steps we've covered are shown here. The plaintext encryption process and the cipher decryption process are opposite each other.



IN THE EARLY 1970s...

LUCIFER INVOLVED A BLOCK LENGTH OF 64 BITS AND A KEY SIZE OF 112 BITS.



## 3-DES ENCRYPTION AND AES ENCRYPTION

BIG BROTHER, DO  
YOU UNDERSTAND  
DES NOW?

BARELY...

BUT ONCE I REVIEW IT...

I'LL UNDOUBTEDLY  
MASTER IT!

YEAH!

THAT'S THE SPIRIT!

SEE?

PAT  
PAT


FIRST YOU'LL HAVE  
TO DEEPEN YOUR  
UNDERSTANDING OF THE  
SIMPLIFIED VERSION,  
SO KEEP AT IT!

BLUSH

BUT CAN DES STILL  
BE USED SAFELY  
NOW? IT'S BEEN  
AROUND SINCE  
THE 1970s!



YES... DES HAS BECOME INSECURE DUE TO ADVANCEMENT IN COMPUTERS AND NOW CAN BE EASILY CRACKED.



### THE DRAWBACKS OF DES

- **SHORT KEY SIZE:** THE SHORTER THE KEY, THE SMALLER THE NUMBER OF POSSIBLE KEYS AVAILABLE AND THE MORE FEASIBLE IT IS TO UNCOVER THE SECRET KEY.
- **OUTDATED DESIGN:** THE STRUCTURAL DESIGN OF DES IS CONSIDERED OBSOLETE AND IS NO LONGER USED. NONRIGOROUS IMPLEMENTATIONS OF CRYPTOGRAPHY COMMONLY APPEAR ON THE MARKET AND ARE EASIER TO CRACK THAN ALREADY ESTABLISHED ONES.

HOW IS IT DONE?

GLOW

CAN EVEN I DO IT?




ABSOLUTELY...

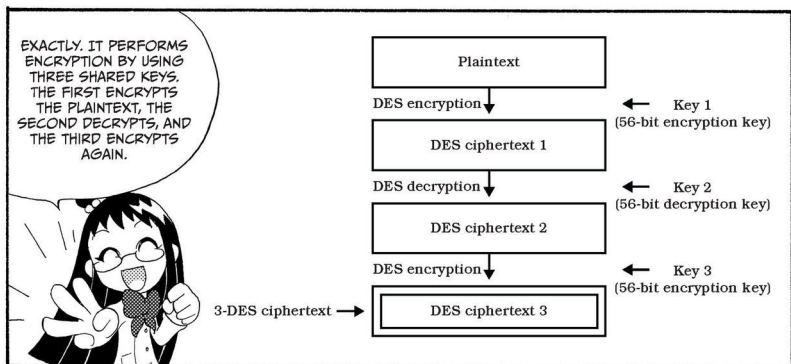
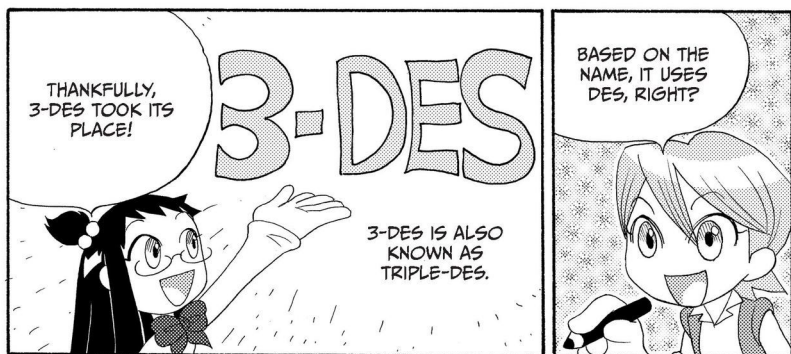
IMPOSSIBLE.



HERE ARE A FEW METHODS FOR CRACKING A BLOCK CIPHER!



|                                    |                                                                                                            |
|------------------------------------|------------------------------------------------------------------------------------------------------------|
| <b>Exhaustive search algorithm</b> | Searching for the key by trying every possible key (brute forcing)                                         |
| <b>Differential cryptanalysis</b>  | Searching for vulnerabilities in the cipher by looking for any correlation between its inputs and outputs  |
| <b>Linear cryptanalysis</b>        | Searching for vulnerabilities in the cipher by looking for mathematical weaknesses in the cipher structure |



WHEN THE FIRST ENCRYPTION  
AND DECRYPTION ROUNDS  
USE THE SAME KEY...

THE RESULT IS THE SAME  
AS DES NO MATTER WHAT  
KEY IS USED FOR THE LAST  
ENCRYPTION. THAT'S BECAUSE  
CIPHERTEXT 2 WOULD END UP  
THE SAME AS THE PLAINTEXT.  
IN OTHER WORDS, YOU'RE  
ONLY ENCRYPTING THE  
PLAINTEXT ONCE.

IF THE FIRST TWO  
KEYS ARE IDENTICAL,  
THE REAL KEY SIZE  
IS 56 BITS.

IF YOU USE THE SAME  
KEY IN BOTH ROUNDS  
OF ENCRYPTION...

AND A DIFFERENT KEY FOR  
DECRYPTION, YOU END  
UP WITH A KEY SIZE OF  
 $56 \times 2 = 112$  BITS!

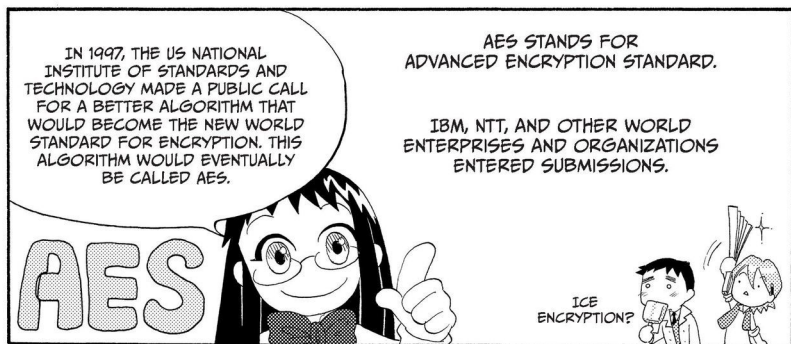
112 bits



THIS 3-DES FORMULA IS KNOWN AS EDE  
(ENCRYPT-DECRYPT-ENCRYPT) MODE.

168 bits!

IF YOU USE DIFFERENT  
KEYS THROUGHOUT, YOU  
END UP WITH A KEY SIZE  
OF  $56 \times 3 = 168$  BITS.



## AN OUTLINE OF AES

In 2000, Rijndael was renamed AES and published as a Federal Information Processing Standard (FIPS). The name Rijndael is derived from the names of its developers, Joan Daemen and Vincent Rijmen, researchers at Belgium's Catholic University of Leuven.

As shown in the following table, in AES there are three possible key sizes.

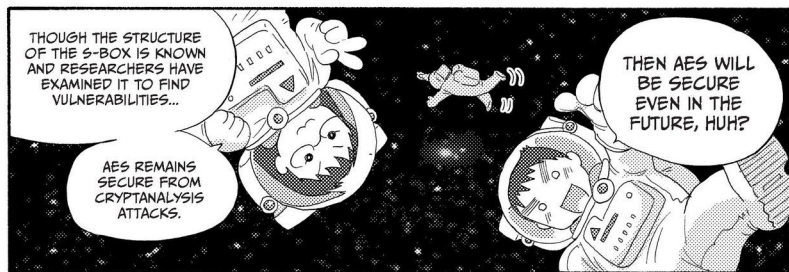
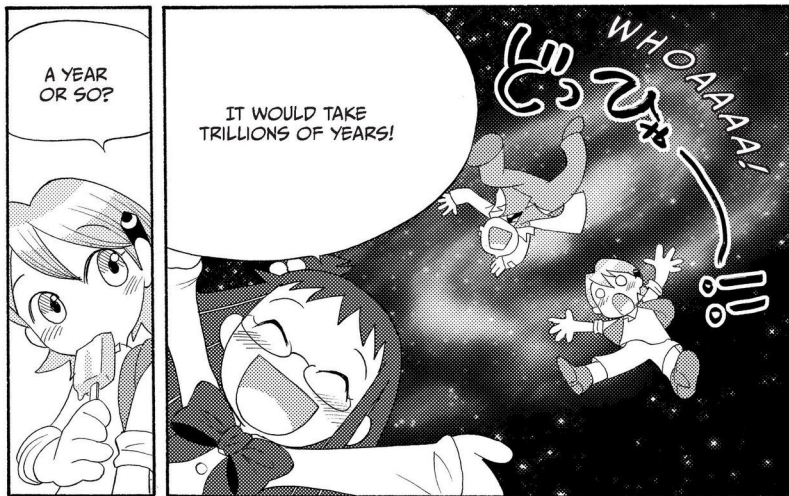
| Type    | Key size (bits) | Block size (bits) | Number of rounds |
|---------|-----------------|-------------------|------------------|
| AES-128 | 128             | 128               | 10               |
| AES-192 | 192             | 128               | 12               |
| AES-256 | 256             | 128               | 14               |

The encryption strength increases with respect to the key size and the number of operations required. The block cipher structure of AES doesn't use a Feistel network but instead uses an SPN (substitution-permutation network). To give a superficial explanation of what an SPN is, imagine the substitution portion as an XOR operation that is performed on each block and subkey while the permutation is some function that simply scrambles the input bits. These two are performed simultaneously, usually for some number of rounds (just as in a Feistel network).

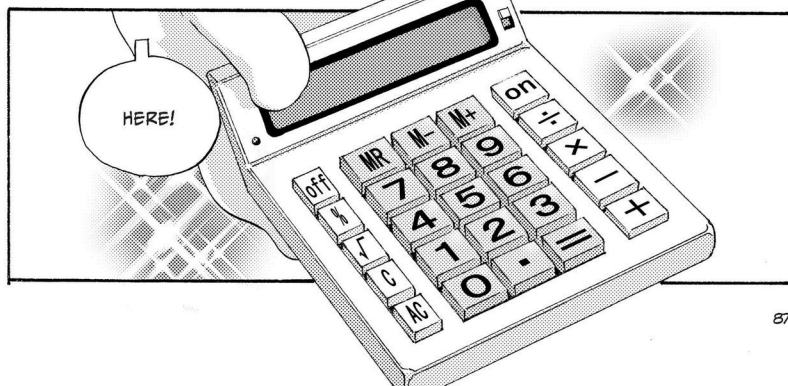
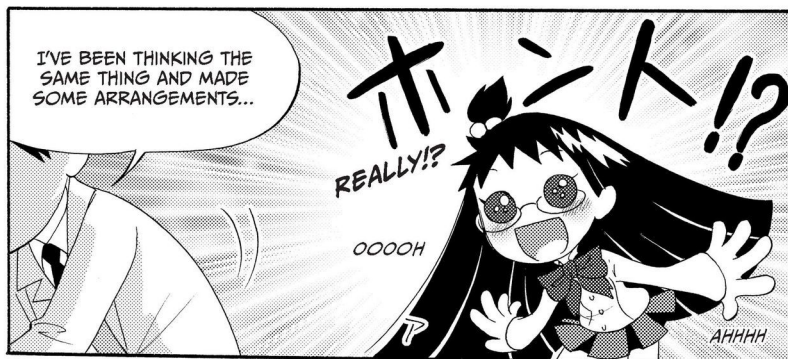
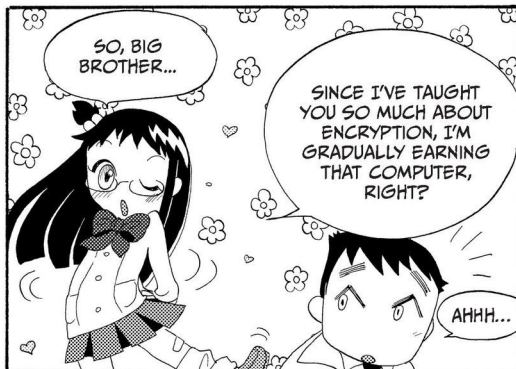
From here on out, we'll leave DES behind and discuss AES.













# SIMPLIFIED DES ENCRYPTION AND DECRYPTION

How are encryption and decryption performed in DES? We will work on a reduced version of the DES algorithm to get an idea of what real DES is like. The adaptation of DES that we'll use here to help you solidify your understanding of the concepts imitates DES but with fewer steps and smaller block sizes (input/output sizes).

## CONVERTING DATA INTO BINARY

Because modern-day algorithms—including DES and others—deal with binary data, we need to be able to convert plaintext messages composed of letters and numbers into binary. Table 2-1 illustrates 16 characters converted to 4-bit binary numbers.

TABLE 2-1: CHARACTERS REPRESENTED IN BINARY

| Characters          | Binary |
|---------------------|--------|
| A                   | 0000   |
| B                   | 0001   |
| C                   | 0010   |
| D                   | 0011   |
| E                   | 0100   |
| F                   | 0101   |
| G                   | 0110   |
| H                   | 0111   |
| I                   | 1000   |
| J                   | 1001   |
| K                   | 1010   |
| L                   | 1011   |
| M                   | 1100   |
| N                   | 1101   |
| O                   | 1110   |
| discarded character | 1111   |

The encoding in Table 2-1 is a simple conversion we will use for the purposes of this book and is not based on any particular encoding used in cryptography. Likewise, the following conversions do not reflect actual binary representations of letters and numbers done in practice (which we saw earlier with ASCII encoding), but rather serve as simple examples of how characters may be converted to binary numbers.

## GENERATING A DES CIPHERTEXT

In DES encryption, a block is 64 bits. However, for the purposes of this book, we will use an 8-bit block in our reduced DES cipher example, as shown in Figure 2-1. Our reduced DES cipher algorithm is composed of two steps—a *key schedule* (which generates a larger key from the key we use) and actual encryption using the generated keys. DES is not the only block cipher to require a key schedule procedure. It is common for block ciphers to perform a key schedule prior to or in parallel with encryption and decryption.

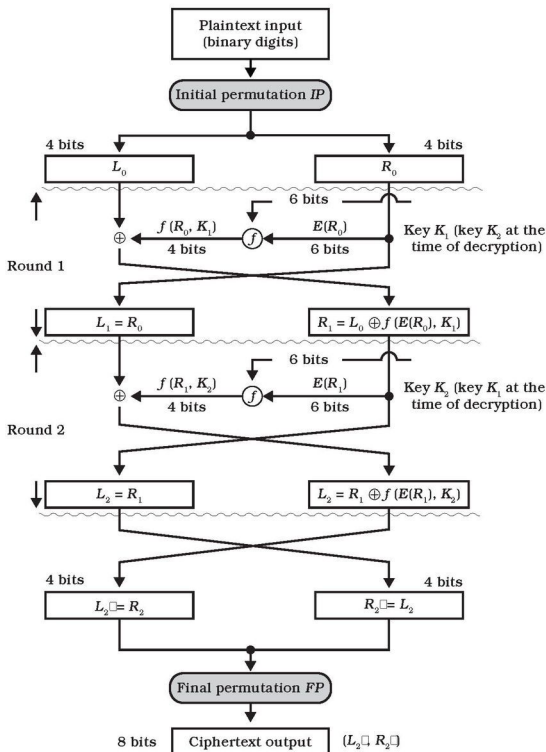


Figure 2-1: Simplified process of generating a DES ciphertext (excluding the key scheduler)

Though it is not shown in Figure 2-1, the plaintext we want to encrypt is first converted into 0s and 1s, using Table 2-1, so that we can work with binary data.

The two binary keys we'll use to perform reduced DES are as follows:

$$K_1 = (110001)_2, K_2 = (111000)_2$$

NOTE: The notations  $( )_2$  and  $( )_{10}$  represent binary digits and decimal numbers, respectively.

These keys have already been generated through the key schedule. We'll discuss how a key is generated through the key schedule later; for now, let's focus on the encryption process of truncated DES.

We'll encrypt the character string "MC" into a ciphertext using the truncated version of DES, so we'll start by expressing these two inputted characters as a block of 8 bits. Based on Table 2-1, "MC" in binary is expressed as 11000010.

## PERFORM THE INITIAL PERMUTATION

We first convert the plaintext to binary, and then we perform the initial permutation on the binary data according to Table 2-2. *IP* permutes the 8-bit binary data by rearranging the input bit positions.

TABLE 2-2: INITIAL PERMUTATION *IP*

|                                              |       |       |       |       |       |       |       |       |
|----------------------------------------------|-------|-------|-------|-------|-------|-------|-------|-------|
| Input bits (original position)               | $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ | $b_6$ | $b_7$ | $b_8$ |
| Output bits (using original input positions) | $b_2$ | $b_4$ | $b_6$ | $b_8$ | $b_1$ | $b_3$ | $b_5$ | $b_7$ |

In Table 2-2, the subscript of each bit indicates its original position, so these numbers are in order in the first row. Then the bit is translated, so for example, the first bit of input data ( $b_1$ ) is transposed to be the fifth bit of output data, and the second bit of input data ( $b_2$ ) is transposed to be the first bit of the output data. Figure 2-2 shows "MC", represented in binary, undergoing the initial permutation.

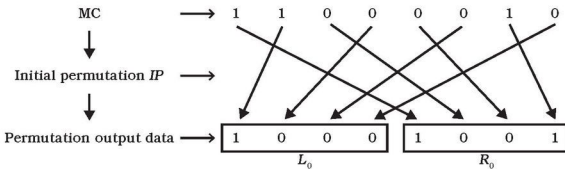


Figure 2-2: Initial permutation of the plaintext "MC"



## SEPARATE DATA INTO HIGHER- AND LOWER-ORDER BITS

We separate the permutation's output data into higher-order and lower-order bits. The higher-order bits are denoted  $L_0$ , and the lower-order bits are denoted  $R_0$ . Based on Figure 2-2, this yields the following:

$$L_0 = (1000)_2$$

$$R_0 = (1001)_2$$

## FEISTEL ROUNDS VARIANT

Now we begin going through rounds in a Feistel network. The Feistel network performs a specific set of steps that “mix bits” to create a new  $L$  and  $R$ . Since we're using a reduced form of DES, we'll mix the bits using a less rigorous method than with real DES. (In real DES, the network mixes the bits in a way that ensures pseudorandomness.) The following represents what a round of the Feistel network looks like in notation form:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(K_i, R_{i-1})$$

Each generation of a new  $L$  and  $R$  is a round, and  $i$  is equal to the round number. The process represented in Figure 2-2 is one round.

$L_1$  will be equal to  $R_0$ , but the process to derive  $R_1$  requires much more work;  $R_0$  and key  $K_1$  are input into function  $f$ . In our reduced DES, there will be only 2 rounds, but real DES has 16 rounds.

Let's walk through how  $f(K_1, R_0)$  works.

### ROUND 1

The first operation in  $f$  is to apply the expansion function  $E$  from Table 2-3 on  $R_0$ .

TABLE 2-3: EXPANSION FUNCTION  $E$

|                                              |       |       |       |       |       |       |
|----------------------------------------------|-------|-------|-------|-------|-------|-------|
| Input bits (original position)               | $b_1$ | $b_2$ | $b_3$ | $b_4$ |       |       |
| Output bits (using original input positions) | $b_3$ | $b_4$ | $b_1$ | $b_2$ | $b_3$ | $b_4$ |

This means that we duplicate the third and fourth bits of  $R_0$  and append them to the beginning of  $R_0$  to form a new  $R_0$  that is 6 bits long:

$$R_0 = (1001)_2 \xrightarrow{E} R_0 = (011001)_2$$

We next XOR the expansion permutation  $R_0$  and the key  $K_1 = (110001)_2$ , as shown in Figure 2-3.

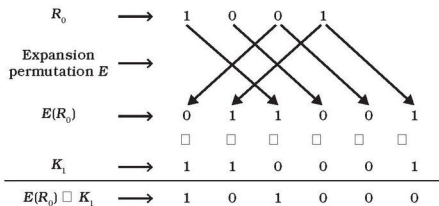


Figure 2-3: Visual representation of XORing  $E(R_0)$  and  $K_1$

The process shown in Figure 2-3 can be represented by the following equation:

$$E(R_0) \oplus K_1 = (011001)_2 \oplus (110001)_2 = (101000)_2$$

Now we put  $R_0 \oplus K_1$  through the substitution box (S-box) in Table 2-4, which compresses and substitutes the bits from  $R_0 \oplus K_1$  to create a 4-bit output. We'll label the bits from  $R_0 \oplus K_1$  as  $b_1b_2b_3b_4b_5b_6$ .

TABLE 2-4: SUBSTITUTION BOX (S-BOX)

|             |   | Column numbers |    |    |   |    |    |    |    |    |    |    |    |    |    |    |    |
|-------------|---|----------------|----|----|---|----|----|----|----|----|----|----|----|----|----|----|----|
|             |   | 0              | 1  | 2  | 3 | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
| Row numbers | 0 | 14             | 4  | 13 | 1 | 2  | 15 | 11 | 8  | 3  | 10 | 6  | 12 | 5  | 9  | 0  | 7  |
|             | 1 | 0              | 15 | 7  | 4 | 14 | 2  | 13 | 1  | 10 | 6  | 12 | 11 | 9  | 5  | 3  | 8  |
|             | 2 | 4              | 1  | 14 | 8 | 13 | 6  | 2  | 11 | 15 | 12 | 9  | 7  | 3  | 10 | 5  | 0  |
|             | 3 | 15             | 12 | 8  | 2 | 4  | 9  | 1  | 7  | 5  | 11 | 3  | 14 | 10 | 0  | 6  | 13 |

In the S-box shown in Table 2-4, four substitutions are available and labeled with row numbers starting with 0 and going up to 3. We determine which substitution to use based on the value of the bits  $b_1b_2b_3b_4b_5b_6$ . We convert the first and last bits,  $b_1$  and  $b_6$ , to their base-10 equivalent and use those two numbers to determine which row of the S-box we use. We then also convert the remaining bits,  $b_2b_3b_4b_5$ , to base 10 to determine which column we will use.

For example, in  $(101000)_2$ , we select row 2 because  $b_1$  is 1 and  $b_6$  is 0 and  $(10)_2 = (2)_{10}$ . Then, bits  $b_2b_3b_4b_5$  are also converted to base 10 to determine the column number we select. Since  $(0100)_2 = (4)_{10}$ , we select column 4 in this case, which contains the value  $(13)_{10}$  (the position that is boxed in Table 2-4). After we find the corresponding table entry, we convert it into binary:  $(1101)_2$ .

The resulting binary,  $(1101)_2$ , then undergoes the permutation  $PS$  (Table 2-5).

TABLE 2-5: PERMUTATION  $PS$

| Input bits (original position)               | $b_1$ | $b_2$ | $b_3$ | $b_4$ |
|----------------------------------------------|-------|-------|-------|-------|
| Output bits (using original input positions) | $b_3$ | $b_4$ | $b_1$ | $b_2$ |

When the permutation  $PS$  from Table 2-5 is applied to our working example, the process looks like Figure 2-4.

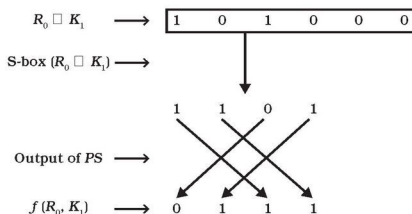


Figure 2-4: The calculation process of  $PS$

The result is  $(1111)_2$ . We've now finished the permutation process. That is to say, we've done all the steps involved in the function  $f$ . All of these steps together are written with this notation:

$$f(K_1, R_0) = (0111)_2$$

This is the last step in the first round of the Feistel network (see Figure 2-1). In this step, we switch the sides of the bits to create  $L_1$  and  $R_1$ . The four lower-order bits on the right side ( $R_0$ ) that we got in "Separate Data into Higher- and Lower-Order Bits" on page 92 are assigned to  $L_1$ , so that they become the higher-order bits.  $L_0$  is assigned to  $R_1$  and becomes the lower-order bits. We can summarize the flip by assigning  $L_1$  to the equation that produced  $R_0$  and assigning  $R_1$  to the equation in "Perform the Initial Permutation" on page 91. Using the equation from "Feistel Rounds Variant" on page 92, we will do the following:

$$\begin{aligned} L_1 &= R_0 = (1001)_2 \\ R_1 &= L_0 \oplus f(K_1, R_0) = (1000)_2 \oplus (0111)_2 = (1111)_2 \end{aligned}$$

In order to complete a second round, we repeat these steps on  $R_1$  and  $L_1$ . Since the steps in round 2 are identical to those in round 1, we'll only show the calculations involved, without detailed explanations.

## ROUND 2

The expansion function from Table 2-3 is applied to  $R_1$ :

$$R_1 = (1111)_2 \xrightarrow{E} \emptyset R_1 = (111111)_2$$

We then XOR  $R_1$  with key  $K_2 = (111000)_2$ :

$$R_1 \oplus K_2 = (111111)_2 \oplus (111000)_2 = (000111)_2$$

Then we use the permutation in Table 2-5 to rearrange the position of the bits, resulting in this:

$$(0100)_2 \oplus (0001)_2$$

We'll apply the equation we use to express XORing  $R_1$  and  $K_2$  and process the result with the S-box:

$$f(K_2, R_1) = (0001)_2$$

For the final step of round 2 (see Figure 2-1 again), we will do the following:

$$L_2 = R_1 = (1111)_2$$

$$R_2 = L_1 \oplus f(K_2, R_1) = (1001)_2 \oplus (0001)_2 = (1000)_2$$

## SWAP THE HIGHER- AND LOWER-ORDER BITS

After round 2 is completed, we swap the higher-order bits  $L_2$  and lower-order bits  $R_2$  again to get  $L_2'$  and  $R_2'$ :

$$L_2' = R_2 = (1000)_2$$

$$R_2' = L_2 = (1111)_2$$

This swap is shown in Figure 2-5.

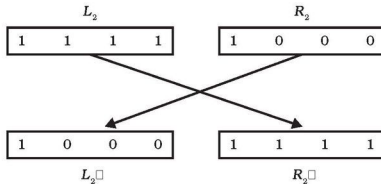


Figure 2-5: Swapping the higher- and lower-order bits

## PERFORM THE FINAL PERMUTATION

Based on the final permutation  $FP$  in Table 2-6, we transpose each bit to a new position.

TABLE 2-6: FINAL PERMUTATION  $FP$ 

| Input bits (original position)               | $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ | $b_6$ | $b_7$ | $b_8$ |
|----------------------------------------------|-------|-------|-------|-------|-------|-------|-------|-------|
| Output bits (using original input positions) | $b_5$ | $b_1$ | $b_6$ | $b_2$ | $b_7$ | $b_3$ | $b_8$ | $b_4$ |

For example, if you look at Table 2-2, you'll see that the fifth input bit ( $b_5$ ) becomes the seventh bit in the output. When the seventh bit ( $b_7$ ) is processed in Table 2-6, it becomes the fifth bit in the output, which means the bit reverts to its initial position. Figure 2-6 shows the initial and final permutations.

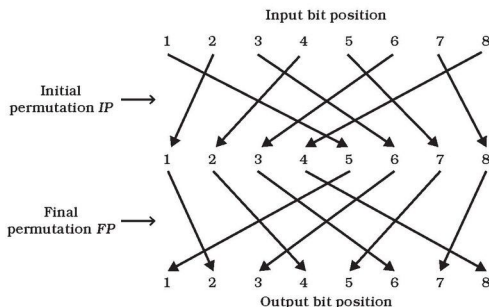


Figure 2-6: The process of putting an 8-bit block through IP and FP. The initial permutation IP and final permutation FP are inverses of one another.

The process shown in Table 2-6 results in  $L_2''$  and  $R_2''$ , which when combined result in our 8-bit ciphertext:

$$L_2'' = (1000)_2$$

$$R_2'' = (1111)_2$$

The resulting ciphertext is  $(11101010)_2$ , as shown in Figure 2-7.

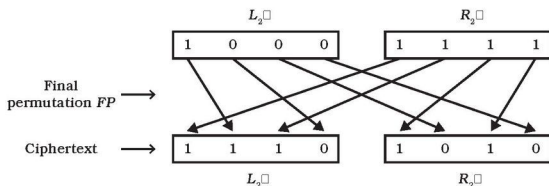


Figure 2-7: The resulting ciphertext from the final permutation FP



## DECRYPTING A DES CIPHERTEXT

Now, we'll try to decrypt the DES ciphertext from Figure 2-7 into its original plaintext. For decryption, you apply the same steps shown in Figure 2-1.

### PERFORM THE INITIAL PERMUTATION

The ciphertext is made up of its left and right sides,  $L_2''$  and  $R_2''$ . In the initial permutation, both sides of the ciphertext's bits (11101010) were rearranged according to the initial permutation chart in Table 2-2.

### SEPARATE DATA INTO HIGHER- AND LOWER-ORDER BITS

The permuted bits are separated into four higher-order bits (left side)  $L_0$  and four lower-order bits (right side)  $R_0$ . These bits are the same as  $L_2'$  and  $R_2'$ . In equation form, this is the opposite of the final permutation when encrypting with DES:

$$L_0 = (1000)_2 (= L_2')$$

$$R_0 = (1111)_2 (= R_2')$$

Figure 2-8 shows the initial permutation and separation processes together.

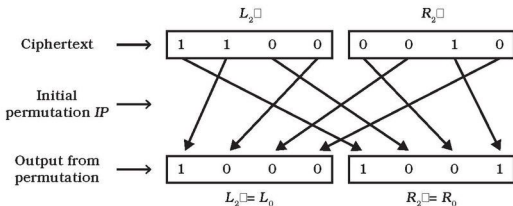


Figure 2-8: Output after applying the initial permutation IP to the ciphertext

### ROUND 1

The first round of decryption inverts the last round of encryption, so we start by applying the expansion function  $E$  from Table 2-3 on  $R_0$ . The expansion function duplicates the first two bits of  $R_0$  and attaches them to the end of  $R_0$ :

$$R_0 = (1111)_2 \xrightarrow{E} R_0 = (111111)_2$$

We calculate the XOR of  $R_0$  and key  $K_2$ . As a reminder,  $K_2$  is  $(111000)_2$ .

$$R_0 \oplus K_2 = (111111)_2 \oplus (111000)_2 = (000111)_2$$

We then compress and substitute the result from the previous step  $(000111)_2$  using the S-box in Table 2-4. We select the first and last bits of the result,  $(000111)_2$ ; combine those two bits into one binary number,  $(01)_2$ ;

and convert the number to decimal,  $(01)_2 = (1)_{10}$ . We then convert the remaining four bits from the original binary number  $(0011)_2$  to decimal,  $(0011)_2 = (3)_{10}$ . In Table 2-4, we find where the two decimal values intersect. The column number is 3 and the row is 1, so the intersecting value is  $(4)_{10}$  (the position that is circled in Table 2-4). The intersecting value is converted into binary to produce  $(0100)_2$ . Then we use the permutation  $PS$  from Table 2-5 on this binary number, so  $(0100)_2$  becomes  $(0001)_2$ . Finally, this function can be expressed as follows:

$$f(K_2, R_0) = (0001)_2$$

So far, we've found the output for the four higher-order bits  $L_1$  and calculated  $f(K_2, R_0) = (0001)_2$ . With this information, we can also derive  $R_1$ . We first take  $L_1$  and set it equal to  $R_0$ . Then, we derive  $R_1$  by XORing  $L_0$  and  $f(K_2, R_0)$ . We'll find the output for the four lower-order bits  $R_1$  as follows:

$$\begin{aligned} L_1 &= R_0 = (1111)_2 \\ R_1 &= L_0 \oplus f(K_2, R_0) = (1000)_2 \oplus (0001)_2 = (1001)_2 \end{aligned}$$

Now that we have finished the first round, we'll perform the second round.

## ROUND 2

As we did in all the previous rounds, we first find  $f(K_1, R_2)$  by applying the expansion function  $E$  from Table 2-3 on  $R_1$ :

$$R_1 = (1001)_2 \xrightarrow{E} R_1 = (011001)_2$$

We calculate the XOR of  $R_1$  and the key  $K_1$ . The key  $K_1$  is equal to  $(110001)_2$ .

$$R_1 \oplus K_1 = (011001)_2 \oplus (110001)_2 = (101000)_2$$

Then we put  $R_1 \oplus K_1$  through the S-box using the permutation  $PS$  from Table 2-5. Doing so rearranges the position of the binary number's bits so that it becomes the following:

$$(1101)_2 \oplus (0111)_2$$

Finally, this whole process is expressed in equation form as this:

$$f(K_1, R_2) = (0111)_2$$

In Figure 2-12, the output of round 2 is the four higher-order bits (left side)  $L_2$  and four lower-order bits (right side)  $R_2$ , which are found using the following equations:

$$\begin{aligned} L_2 &= R_1 = (1001)_2 \\ R_2 &= L_1 \oplus f(K_1, R_1) = (1111)_2 \oplus (0111)_2 = (1000)_2 \end{aligned}$$

$L_2$  is set equal to  $R_1$ , which in this case is the binary value 1001.  $R_2$  is found by XORing  $L_1$  and  $f(K_1, R_1)$ .

In the final stage of the process in Figure 2-1, we swap the higher-order bits,  $L_2$ , and lower-order bits,  $R_2$ , to find  $L_2'$  and  $R_2'$ , as seen in Figure 2-9.

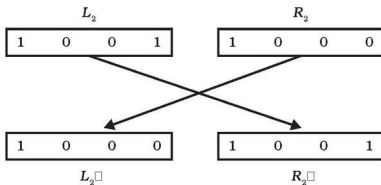


Figure 2-9: Swapping the left and right bits

This swap is represented by the following equations:

$$L_2' = R_2 = (1000)_2$$

$$R_2' = L_2 = (1001)_2$$

$L_2''$  and  $R_2''$  are found by performing the final permutation  $FP$  from Table 2-6 on  $L_2'$  and  $R_2'$ . Using Table 2-1,  $L_2''$  is then converted from binary to the character "M", and  $R_2''$  is converted to "C".

$$L_2'' = (1100)_2$$

$$R_2'' = (0010)_2$$

You can see this process in Figure 2-10.

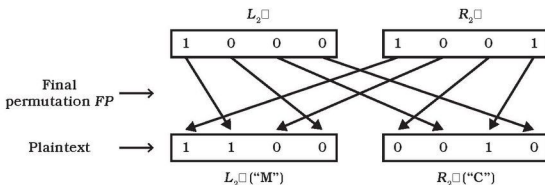


Figure 2-10: The final permutation performed on the bits of  $L_2''$  and  $R_2''$

The plaintext is 1100 0010, which is the character string "MC".

We can see that the resulting 8-bit output data corresponds with the original plaintext, so the ciphertext has been decrypted. As you can see, decryption is the same process as encryption, just with the order of keys reversed. The relationship between encryption and decryption processes is shown in Figure 2-11.

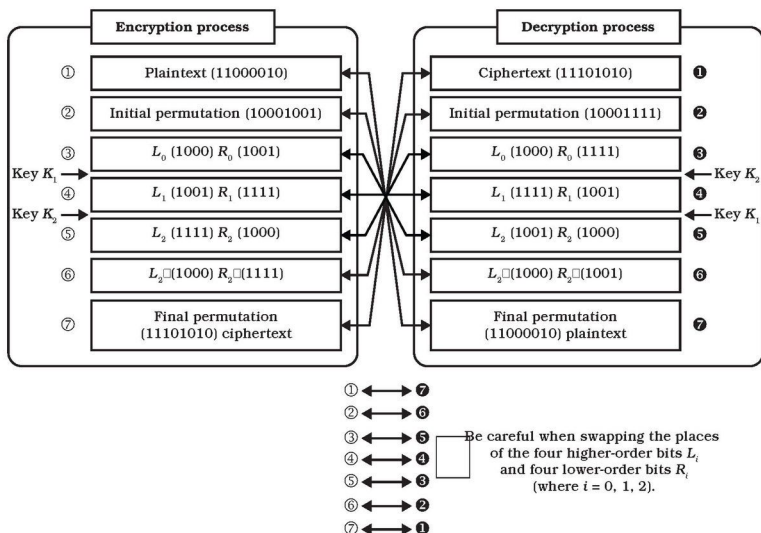


Figure 2-11: Relationship between the encryption and decryption processes

## GENERATING DES KEYS

Now that we have seen how encryption and decryption are performed using reduced DES, let's discuss how keys are generated from the key schedule. Reduced DES requires an input of an 8-bit key  $K_0$ , and the key given by the user generates two 8-bit keys, which are then used in the encryption and decryption process.

$$K_0 = (10011001)_2$$

The key schedule, which is the procedure for generating the keys  $K_1$  and  $K_2$ , is shown in Figure 2-12.

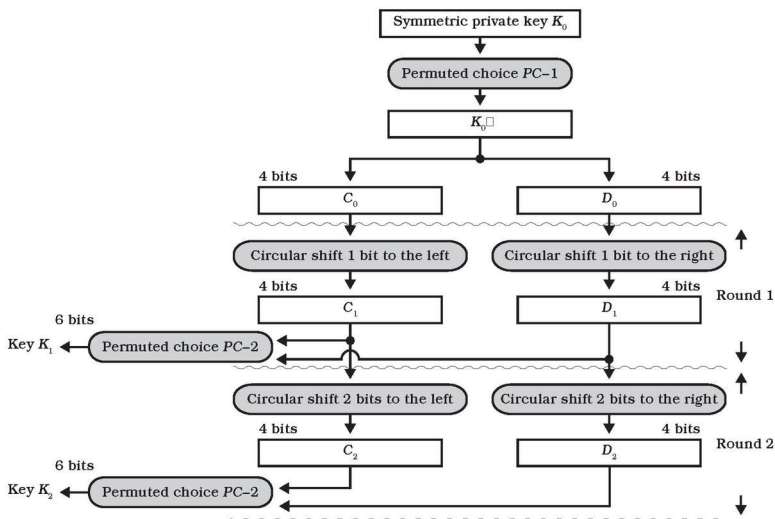


Figure 2-12: Generating encryption keys and decryption keys

### PERFORM PERMUTED CHOICE PC-1

The symmetric key  $K_0$  is permuted according to PC-1.

$$K_0' = (00110101)_2$$

The bits are transposed, as shown in Table 2-7.

TABLE 2-7: PERMUTED CHOICE PC-1

|                                              | 4 higher-order bits |       |       |       | 4 lower-order bits |       |       |       |
|----------------------------------------------|---------------------|-------|-------|-------|--------------------|-------|-------|-------|
| Input bits (original position)               | $b_1$               | $b_2$ | $b_3$ | $b_4$ | $b_5$              | $b_6$ | $b_7$ | $b_8$ |
| Output bits (using original input positions) | $b_3$               | $b_6$ | $b_4$ | $b_8$ | $b_7$              | $b_5$ | $b_2$ | $b_1$ |



When the transposition in Table 2-7 is applied to  $K_0$ , we acquire  $K'_0$ , as shown in Figure 2-13.

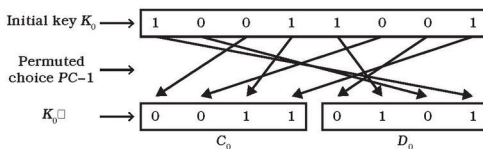


Figure 2-13: Permutated choice PC-1

Here,  $K'_0$  is divided into  $C_0$ , which is made up of the four higher-order bits, and  $D_0$ , the key's four lower-order bits, and is expressed as follows:

$$C_0 = (0011)_2$$

$$D_0 = (0101)_2$$

## ROTATE THE BITS

$C_0$  and  $D_0$  are each rotated one bit to the left, as Figure 2-14 shows, and the results are expressed as  $C_1$  and  $D_1$ . During round 1, bits are all shifted by one position.

$$C_1 = (0110)_2$$

$$D_1 = (1010)_2$$

The bits in  $C_0$  and  $D_0$  are rotated independently to produce  $C_1$  and  $D_1$ , as shown in Figure 2-14.

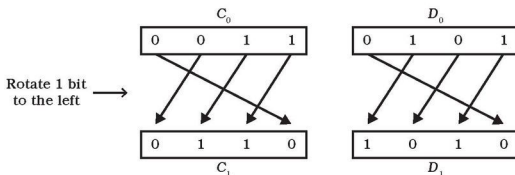


Figure 2-14: Rotating  $C_0$  and  $D_0$  one bit to the left

## PERFORM PERMUTED CHOICE PC-2

Based on permuted choice PC-2,  $C_1$  and  $D_1$  are compressed from 8 bits to 6 bits, yielding the key  $K_1$ , which is used in round 1 of encryption.

$$K_1 = (110001)_2$$

Permuted choice PC-2 is shown in Table 2-8.

TABLE 2-8: PERMUTED CHOICE PC-2

| Position of input bits                  | $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ | $b_6$ | $b_7$ | $b_8$ |
|-----------------------------------------|-------|-------|-------|-------|-------|-------|-------|-------|
| Output labeled with original input bits | $b_7$ | $b_5$ | $b_1$ | $b_8$ | $b_6$ | $b_2$ |       |       |

Figure 2-15 shows the results of applying permuted choice PC-2 from Table 2-8 to  $C_1$  and  $D_1$ .

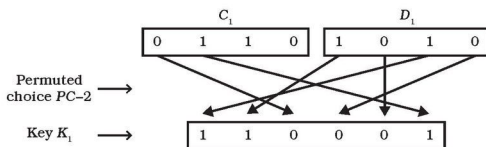


Figure 2-15: Permuted choice PC-2

Repeating the key schedule steps will yield new keys to use for reduced DES.

## ROTATE THE BITS AGAIN

Now we are going to generate  $K_2$ . Each bit of  $C_1$  and  $D_1$  is rotated to the left by two bits, as shown in Figure 2-16.

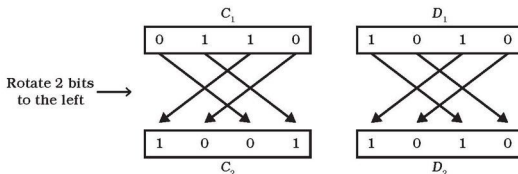


Figure 2-16: Rotating  $C_1$  and  $D_1$  two bits to the left

This results in  $C_2$  and  $D_2$ :

$$C_2 = (1001)_2$$

$$D_2 = (1010)_2$$

## COMPRESS THE BITS

Based on permuted choice PC-2 in Table 2-8,  $C_2$  and  $D_2$  are compressed from a total of 8 bits to 6 bits. This yields key  $K_2$ , which is used in the second round of encryption:

$$K_2 = (111000)_2$$

Figure 2-17 shows how permuted choice PC-2 compresses the bits of  $C_2$  and  $D_2$ .

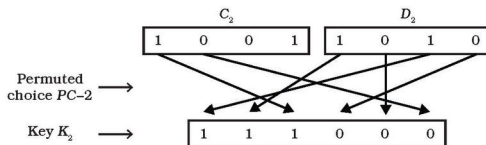


Figure 2-17: Using permuted choice PC-2 to produce  $K_2$

## HOW REDUCED DES DIFFERS FROM REAL DES

We've learned how reduced DES works. As mentioned at the beginning of this chapter, real DES used in practice is more complex.

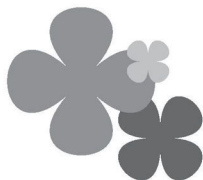
**DES (and its reduced version) is no longer secure and should not be used in any practical implementations.**

Reduced DES is a modified DES that encrypts and decrypts only 8-bit messages using an 8-bit key, while real DES encrypts and decrypts 64-bit messages with a 56-bit key. This means that all of the procedures done in real DES use a different expansion function,  $f$  function, and S-box to accommodate for the size of the inputs. In addition, real DES performs 16 rounds rather than the 2 rounds of the Feistel network. The key schedule is also different, because real DES generates 16 keys and performs only one circular bit shift rather than two.

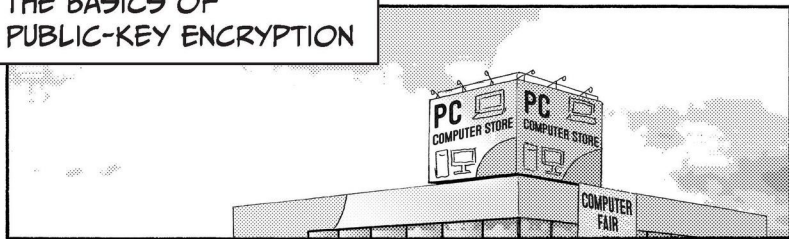
Although we didn't cover real DES, this section should still give you an idea of just how involved, exhaustive, and particular the steps in modern-day encryption schemes are.

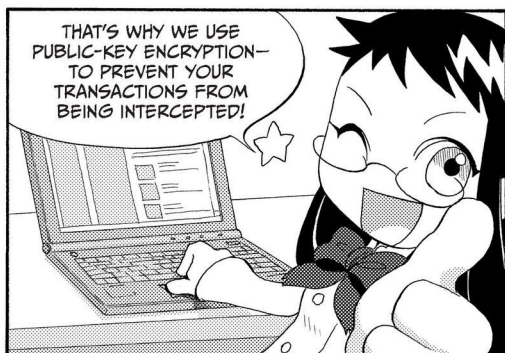
# 3

## PUBLIC-KEY ENCRYPTION

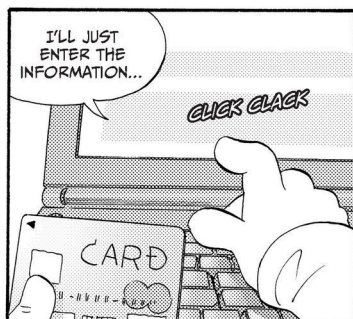


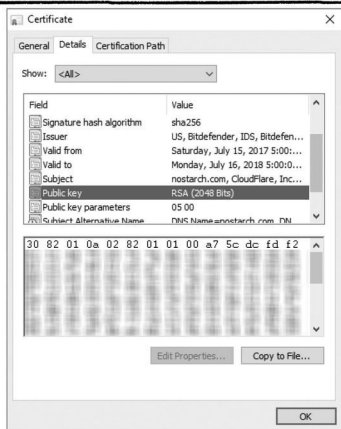
# THE BASICS OF PUBLIC-KEY ENCRYPTION





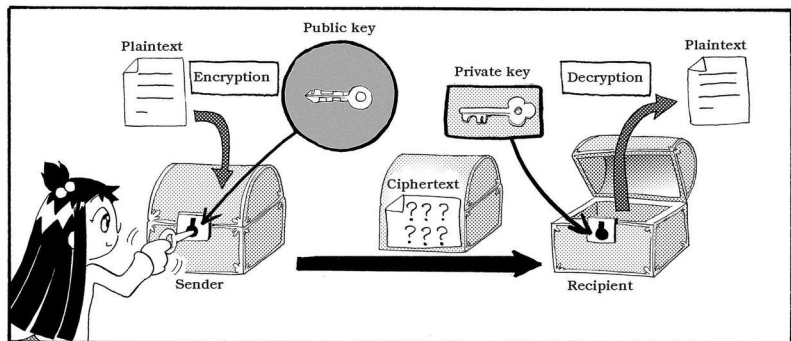
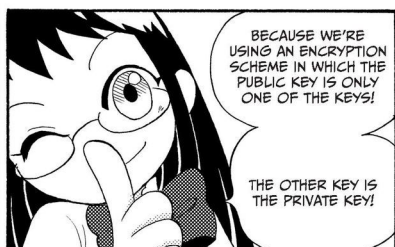


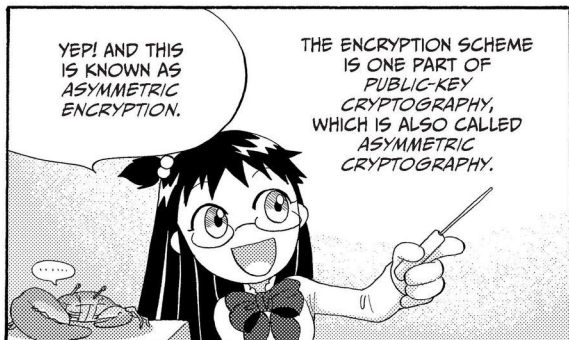
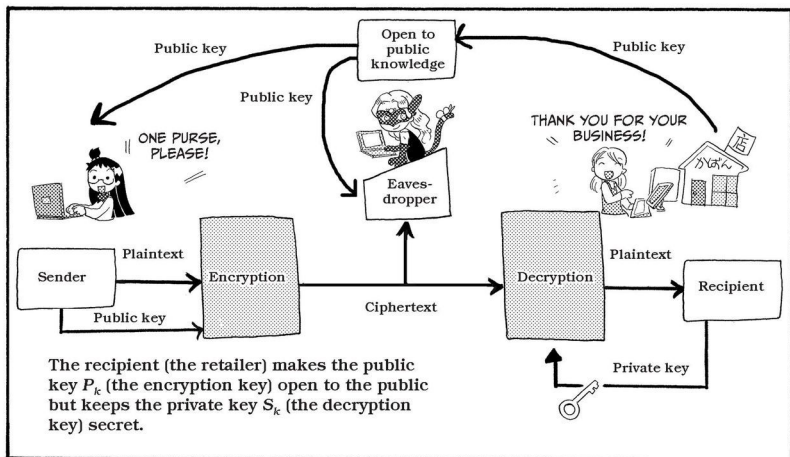




You can view the public key used for encrypting messages to a website you are currently communicating with in your web browser. If you're using Google Chrome, click the lock symbol in the address bar, and then click the Valid option under the Certificate heading. Switch to the Details tab and select Public key. You would also click the lock symbol in Firefox and Safari to view your SSL certificate.











BECAUSE EVERYONE INVOLVED IN THE CORRESPONDENCE HAS ONE PRIVATE KEY AND ONE PUBLIC KEY EACH.

EVERYONE CAN COMMUNICATE WITH A SMALLER TOTAL NUMBER OF KEYS.

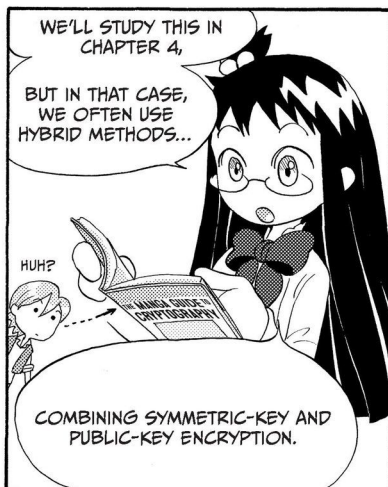
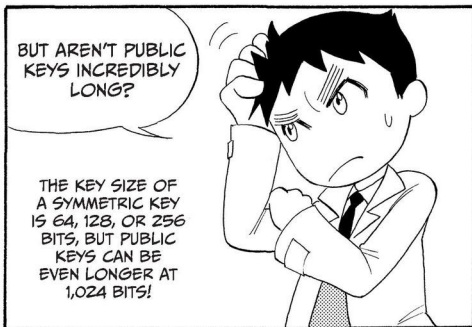
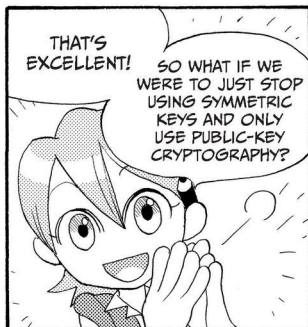
In the public-key encryption scheme, even if  $n$  users were communicating with encryption, the total number of keys involved is only  $2n$ . That's because each user's public key from their key pair is what others will use to encrypt a message to that person. When there are 1,000 users in a symmetric-key scheme, the number of keys is calculated as follows (using the equation on page 61):

$${}_{1,000}C_2 = \frac{1,000 \times (1,000 - 1)}{2}$$

This means that you need 499,500 keys for a symmetric-key scheme. But for public-key encryption, the key calculation is just  $2 \times 1,000$ , which means you need only 2,000 keys.

WOW!







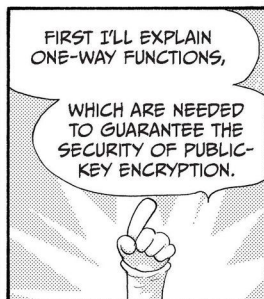
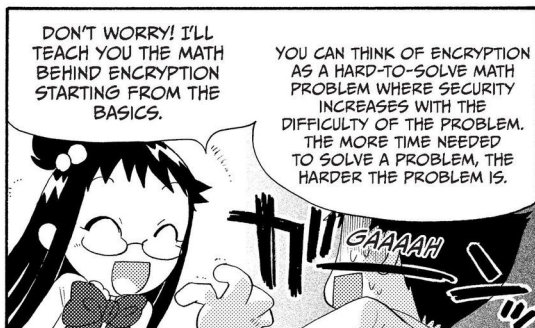
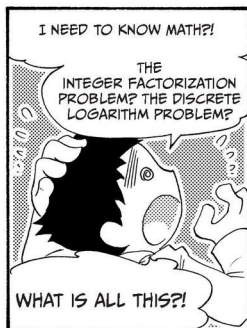


## MATHEMATICAL DEPENDENCIES OF PUBLIC-KEY ENCRYPTION

PUBLIC-KEY ENCRYPTION SCHEMES RELY ON TWO TYPES OF MATHEMATICAL PROBLEMS:

THE INTEGER FACTORIZATION PROBLEM IS USED BY RSA ENCRYPTION, RABIN ENCRYPTION, AND OTHER SCHEMES.

THE DISCRETE LOGARITHM PROBLEM IS USED BY ELGAMAL ENCRYPTION, DSA AUTHENTICATION, AND SO ON.



## ONE-WAY FUNCTIONS

A *one-way function* is a function  $F$  for which it is easy to compute  $F(x)$  for a given input  $x$  but not easy to find the input  $x$  given the calculation of  $F(x)$ . Let's look at an example of a one-way function.

### INTEGER FACTORIZATION PROBLEM

Multiplying two large prime numbers and finding the resulting composite number is simple. On the other hand, attempting to find the original two primes is exceedingly difficult when you are given only the composite number. The process of deriving the original primes based on the composite number is known as the integer factorization problem (explained later in “Prime Numbers and Integer Factorization” on page 120).

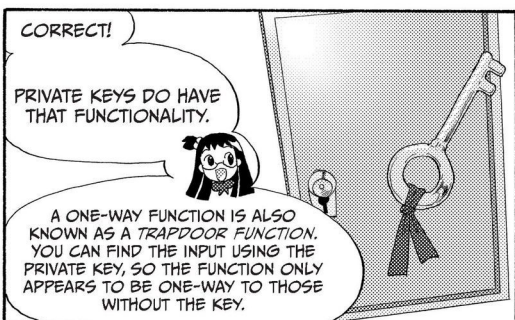
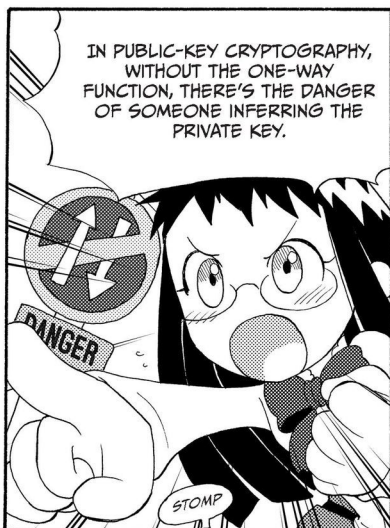
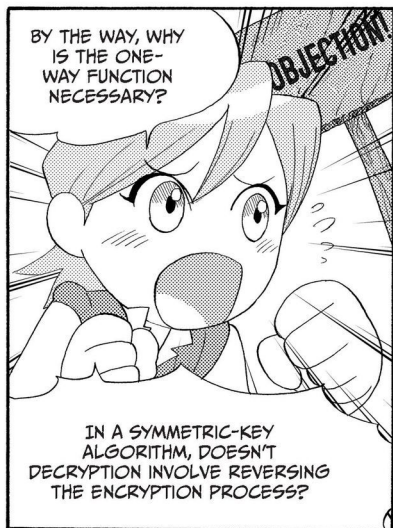
### DISCRETE LOGARITHM PROBLEM

Consider the following equation, which uses modular arithmetic:

$$a^x = y \pmod{p}$$

When you know  $a$  and  $x$ , finding  $y$  is comparatively simple. But if you know  $a$  and  $y$ , finding  $x$ , the logarithm of  $y$ , is extraordinarily difficult. This is the discrete logarithm problem.







If you leave through an automatically locking door, then without a key, you can't get back in. A function with this type of structure is known as a trapdoor function.



Without a key, you can leave the room.



But without a key, you can't enter the room.



If you have a key, you can enter the room.



LET'S TAKE A LOOK AT THE BIRTH OF RSA ENCRYPTION, WHICH IS A PUBLIC-KEY ENCRYPTION SCHEME,

AS WELL AS HOW TO USE THE MATHEMATICS BEHIND IT!



## THE BIRTH OF RSA ENCRYPTION

RSA was introduced in 1977 as the world's first public-key encryption system. Its name derives from the initials of the three American researchers who developed it: Rivest, Shamir, and Adleman.

RSA's strength lies in the difficulty of solving the integer factorization problem developers published in a science magazine in 1977. The problem involves applying integer factorization to a given number to decipher a message.

Here is the 129-digit natural number they presented:

1143816257578888676692357799761466120102182967212  
4236256256184293570693524573389783059712356395870  
5058989075147599290026879543541

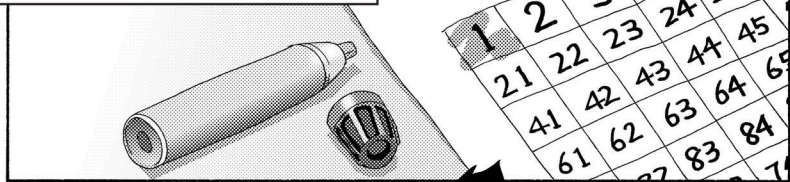
Not until 1994 was the number's integer factorization calculated and the message deciphered. Doing so took some 1,600 computers. It may seem as though 17 years is a long time to find the factors of a number, but one of the RSA developers, Rivest, had anticipated that it would take a thousand years, so the creators were actually surprised the cipher was cracked when it was. Incidentally, the decoded message was "THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE."

The numbers currently used in RSA encryption are more than 300 digits long in base 10 and, even with computers, would take an astronomical amount of time to perform integer factorization on.



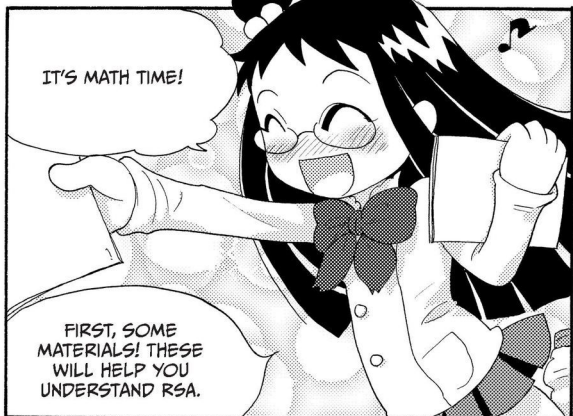


# PRIME NUMBERS AND INTEGER FACTORIZATION



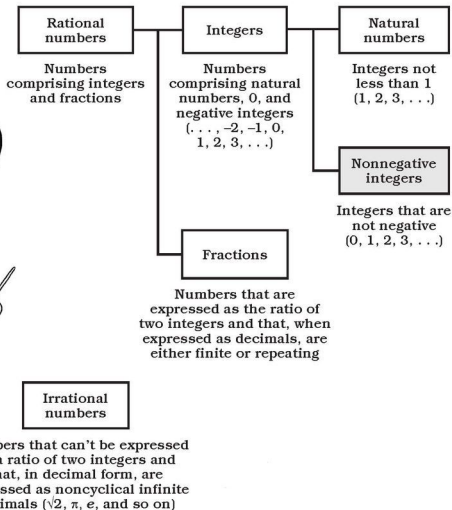
IT'S MATH TIME!

FIRST, SOME  
MATERIALS! THESE  
WILL HELP YOU  
UNDERSTAND RSA.



IN THE MATH BEHIND RSA,  
YOU ONLY HAVE TO WORK  
WITH NATURAL NUMBERS.

THERE AREN'T ANY  
IRRATIONAL NUMBERS  
OR FRACTIONS—  
KEEP THIS IN MIND!



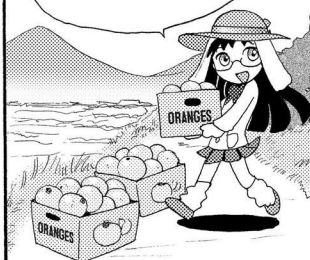
WHAT?  
REALLY?

LET'S  
DO IT!



TIME FOR A  
PROBLEM!

SAY YOU HAVE  
30 ORANGES.



HOW WOULD  
YOU DIVIDE THEM  
EQUALLY AMONG  
A GROUP OF  
CHILDREN, WITH NO  
REMAINDER?

SIMPLE!

HERE'S THE RELATIONSHIP  
BETWEEN THE NUMBER OF  
PEOPLE AND THE NUMBER  
OF ORANGES:

| Number of<br>children | Oranges<br>per child |
|-----------------------|----------------------|
| 1                     | 30                   |
| 2                     | 15                   |
| 3                     | 10                   |
| 5                     | 6                    |
| 6                     | 5                    |
| 10                    | 3                    |
| 15                    | 2                    |
| 30                    | 1                    |

THAT'S IT!

A NUMBER THAT DIVIDES  
PEOPLE, ORANGES, AND  
OTHER THINGS WITH NO  
REMAINDER...

IS KNOWN AS A  
DIVISOR OR A FACTOR.

THE DIVISORS (OR  
FACTORS) OF 30 ARE  
1, 2, 3, 5, 6, 10, 15, AND  
30—EIGHT IN ALL.

AND THEN THERE ARE  
SOME NATURAL NUMBERS  
THAT ARE DIVISIBLE  
BY NOTHING BUT 1 AND  
THEMSELVES—THOSE  
ARE KNOWN AS PRIME  
NUMBERS.

IS 1 A PRIME  
NUMBER?

1

SORRY, 1.

A FUNDAMENTAL RULE  
OF MATHEMATICS IS  
THAT 1 ISN'T A PRIME  
NUMBER.

LET'S TAKE A LOOK  
AT THE PRIME  
NUMBERS UP TO 20.

|    |                                     |              |
|----|-------------------------------------|--------------|
| 2  | Divisible only by itself (2) and 1  | Prime number |
| 3  | Divisible only by itself (3) and 1  | Prime number |
| 4  | Divisible by 2                      | Not prime    |
| 5  | Divisible only by itself (5) and 1  | Prime number |
| 6  | Divisible by 2 and 3                | Not prime    |
| 7  | Divisible only by itself (7) and 1  | Prime number |
| 8  | Divisible by 2 and 4                | Not prime    |
| 9  | Divisible by 3                      | Not prime    |
| 10 | Divisible by 2 and 5                | Not prime    |
| 11 | Divisible only by itself (11) and 1 | Prime number |
| 12 | Divisible by 2, 3, 4, and 6         | Not prime    |
| 13 | Divisible only by itself (13) and 1 | Prime number |
| 14 | Divisible by 2 and 7                | Not prime    |
| 15 | Divisible by 3 and 5                | Not prime    |
| 16 | Divisible by 2, 4, and 8            | Not prime    |
| 17 | Divisible only by itself (17) and 1 | Prime number |
| 18 | Divisible by 2, 3, 6, and 9         | Not prime    |
| 19 | Divisible only by itself (19) and 1 | Prime number |
| 20 | Divisible by 2, 4, 5, and 10        | Not prime    |



$$\begin{aligned}
 4 &= 2^2 = 2 \times 2 \\
 6 &= 2 \times 3 \\
 8 &= 2^3 = 2 \times 2 \times 2 \\
 9 &= 3^2 = 3 \times 3 \\
 10 &= 2 \times 5 \\
 12 &= 2^2 \times 3 = 2 \times 2 \times 3 \\
 14 &= 2 \times 7 \\
 15 &= 3 \times 5 \\
 16 &= 2^4 = 2 \times 2 \times 2 \times 2 \\
 18 &= 2 \times 3^2 = 2 \times 3 \times 3 \\
 20 &= 2^2 \times 5 = 2 \times 2 \times 5
 \end{aligned}$$

INTEGER FACTORIZATION FOR ANY COMPOSITE NUMBER HAS ONLY ONE RESULT. ACCORDING TO THE *UNIQUE FACTORIZATION THEOREM*, EACH FACTORIZATION FOR A COMPOSITE NUMBER IS ALSO UNIQUE TO THAT NUMBER. TO PRESERVE THE INTEGER FACTORIZATION'S UNIQUENESS, WE DON'T CATEGORIZE 1 AS A PRIME NUMBER.

IF 1 WERE PRIME, NUMBERS COULD HAVE MORE THAN ONE FACTORIZATION. FOR EXAMPLE, THE FACTORS OF 6 WOULD BE BOTH  $3 \times 2$  AND  $3 \times 2 \times 1$ .

WHAAAAAT?!

DO WE HAVE TO INSPECT EACH AND EVERY NUMBER TO DETERMINE WHETHER IT'S PRIME?

FORTUNATELY, WE CAN USE A METHOD KNOWN AS THE *SIEVE OF ERATOSTHENES*.

HERE'S HOW IT WORKS!

WHEN A NATURAL NUMBER  $N$  ISN'T DIVISIBLE BY ALL OF THE PRIME NUMBERS LESS THAN  $\sqrt{N}$ , THE NATURAL NUMBER  $N$  IS A PRIME NUMBER.

ERATOSTHENES OF CYRENE



HOW COME?



THINK OF  $N$  AS BEING  
EQUAL TO  $pq$ .

$$p \leq \sqrt{N} \text{ OR } q \leq \sqrt{N}$$

IF  $N$  CAN BE EXPRESSED  
AS THE PRODUCT OF THE  
TWO PRIME NUMBERS  $pq$ ,  
AT LEAST  $p$  OR  $q$  MUST BE  
LESS THAN  $\sqrt{N}$ .

I SEE!

SO IF BOTH  $p$  AND  
 $q$  ARE LARGER THAN  
 $\sqrt{N}$ , THEIR PRODUCT  
WOULD HAVE TO BE  
GREATER THAN  $N$ .

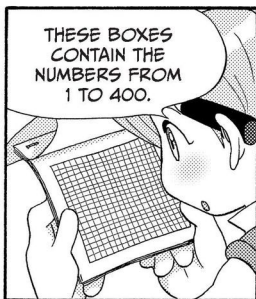
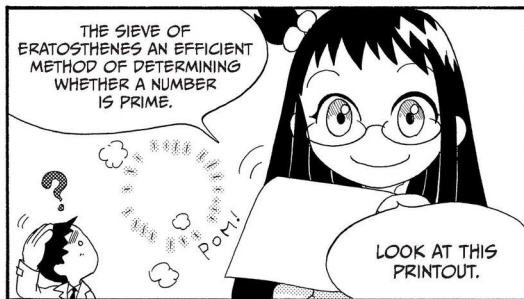
$$p > \sqrt{N} \text{ AND } q > \sqrt{N}$$

$$\downarrow$$
$$pq > N$$

SO WHAT'S ERA...  
WHATEVER IT IS  
YOU SAID?

I'M AN ANCIENT  
GREEK SCHOLAR!

I WAS THE FIRST  
PERSON TO  
CALCULATE THE  
SIZE OF EARTH!



|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20  |
| 21  | 22  | 23  | 24  | 25  | 26  | 27  | 28  | 29  | 30  | 31  | 32  | 33  | 34  | 35  | 36  | 37  | 38  | 39  | 40  |
| 41  | 42  | 43  | 44  | 45  | 46  | 47  | 48  | 49  | 50  | 51  | 52  | 53  | 54  | 55  | 56  | 57  | 58  | 59  | 60  |
| 61  | 62  | 63  | 64  | 65  | 66  | 67  | 68  | 69  | 70  | 71  | 72  | 73  | 74  | 75  | 76  | 77  | 78  | 79  | 80  |
| 81  | 82  | 83  | 84  | 85  | 86  | 87  | 88  | 89  | 90  | 91  | 92  | 93  | 94  | 95  | 96  | 97  | 98  | 99  | 100 |
| 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 |
| 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 |
| 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 |
| 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 180 |
| 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 |
| 201 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 |
| 221 | 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 | 240 |
| 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 | 256 | 257 | 258 | 259 | 260 |
| 261 | 262 | 263 | 264 | 265 | 266 | 267 | 268 | 269 | 270 | 271 | 272 | 273 | 274 | 275 | 276 | 277 | 278 | 279 | 280 |
| 281 | 282 | 283 | 284 | 285 | 286 | 287 | 288 | 289 | 290 | 291 | 292 | 293 | 294 | 295 | 296 | 297 | 298 | 299 | 300 |
| 301 | 302 | 303 | 304 | 305 | 306 | 307 | 308 | 309 | 310 | 311 | 312 | 313 | 314 | 315 | 316 | 317 | 318 | 319 | 320 |
| 321 | 322 | 323 | 324 | 325 | 326 | 327 | 328 | 329 | 330 | 331 | 332 | 333 | 334 | 335 | 336 | 337 | 338 | 339 | 340 |
| 341 | 342 | 343 | 344 | 345 | 346 | 347 | 348 | 349 | 350 | 351 | 352 | 353 | 354 | 355 | 356 | 357 | 358 | 359 | 360 |
| 361 | 362 | 363 | 364 | 365 | 366 | 367 | 368 | 369 | 370 | 371 | 372 | 373 | 374 | 375 | 376 | 377 | 378 | 379 | 380 |
| 381 | 382 | 383 | 384 | 385 | 386 | 387 | 388 | 389 | 390 | 391 | 392 | 393 | 394 | 395 | 396 | 397 | 398 | 399 | 400 |



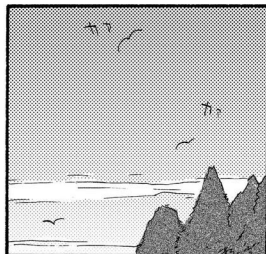


|     | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 21  | 22  | 23  | 24  | 25  | 26  | 27  | 28  | 29  | 30  | 31  | 32  | 33  | 34  | 35  | 36  | 37  | 38  | 39  | 40  |
| 41  | 42  | 43  | 44  | 45  | 46  | 47  | 48  | 49  | 50  | 51  | 52  | 53  | 54  | 55  | 56  | 57  | 58  | 59  | 60  |
| 61  | 62  | 63  | 64  | 65  | 66  | 67  | 68  | 69  | 70  | 71  | 72  | 73  | 74  | 75  | 76  | 77  | 78  | 79  | 80  |
| 81  | 82  | 83  | 84  | 85  | 86  | 87  | 88  | 89  | 90  | 91  | 92  | 93  | 94  | 95  | 96  | 97  | 98  | 99  | 100 |
| 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 |
| 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 |
| 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 |
| 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 180 |
| 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 |
| 201 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 |
| 221 | 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 | 240 |
| 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 | 256 | 257 | 258 | 259 | 260 |
| 261 | 262 | 263 | 264 | 265 | 266 | 267 | 268 | 269 | 270 | 271 | 272 | 273 | 274 | 275 | 276 | 277 | 278 | 279 | 280 |
| 281 | 282 | 283 | 284 | 285 | 286 | 287 | 288 | 289 | 290 | 291 | 292 | 293 | 294 | 295 | 296 | 297 | 298 | 299 | 300 |
| 301 | 302 | 303 | 304 | 305 | 306 | 307 | 308 | 309 | 310 | 311 | 312 | 313 | 314 | 315 | 316 | 317 | 318 | 319 | 320 |
| 321 | 322 | 323 | 324 | 325 | 326 | 327 | 328 | 329 | 330 | 331 | 332 | 333 | 334 | 335 | 336 | 337 | 338 | 339 | 340 |
| 341 | 342 | 343 | 344 | 345 | 346 | 347 | 348 | 349 | 350 | 351 | 352 | 353 | 354 | 355 | 356 | 357 | 358 | 359 | 360 |
| 361 | 362 | 363 | 364 | 365 | 366 | 367 | 368 | 369 | 370 | 371 | 372 | 373 | 374 | 375 | 376 | 377 | 378 | 379 | 380 |
| 381 | 382 | 383 | 384 | 385 | 386 | 387 | 388 | 389 | 390 | 391 | 392 | 393 | 394 | 395 | 396 | 397 | 398 | 399 | 400 |



|     | 2 | 3   | 5   | 7   | 9   | 11  | 13  | 15  | 17  | 19  |
|-----|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 21  |   | 23  | 25  | 27  | 29  | 31  | 33  | 35  | 37  | 39  |
| 41  |   | 43  | 45  | 47  | 49  | 51  | 53  | 55  | 57  | 59  |
| 61  |   | 63  | 65  | 67  | 69  | 71  | 73  | 75  | 77  | 79  |
| 81  |   | 83  | 85  | 87  | 89  | 91  | 93  | 95  | 97  | 99  |
| 101 |   | 103 | 105 | 107 | 109 | 111 | 113 | 115 | 117 | 119 |
| 121 |   | 123 | 125 | 127 | 129 | 131 | 133 | 135 | 137 | 139 |
| 141 |   | 143 | 145 | 147 | 149 | 151 | 153 | 155 | 157 | 159 |
| 161 |   | 163 | 165 | 167 | 169 | 171 | 173 | 175 | 177 | 179 |
| 181 |   | 183 | 185 | 187 | 189 | 191 | 193 | 195 | 197 | 199 |
| 201 |   | 203 | 205 | 207 | 209 | 211 | 213 | 215 | 217 | 219 |
| 221 |   | 223 | 225 | 227 | 229 | 231 | 233 | 235 | 237 | 239 |
| 241 |   | 243 | 245 | 247 | 249 | 251 | 253 | 255 | 257 | 259 |
| 261 |   | 263 | 265 | 267 | 269 | 271 | 273 | 275 | 277 | 279 |
| 281 |   | 283 | 285 | 287 | 289 | 291 | 293 | 295 | 297 | 299 |
| 301 |   | 303 | 305 | 307 | 309 | 311 | 313 | 315 | 317 | 319 |
| 321 |   | 323 | 325 | 327 | 329 | 331 | 333 | 335 | 337 | 339 |
| 341 |   | 343 | 345 | 347 | 349 | 351 | 353 | 355 | 357 | 359 |
| 361 |   | 363 | 365 | 367 | 369 | 371 | 373 | 375 | 377 | 379 |
| 381 |   | 383 | 385 | 387 | 389 | 391 | 393 | 395 | 397 | 399 |





|     |     |    |  |   |     |     |     |     |     |     |  |     |     |     |
|-----|-----|----|--|---|-----|-----|-----|-----|-----|-----|--|-----|-----|-----|
|     | 2   | 3  |  | 5 |     | 7   |     |     | 11  | 13  |  |     | 17  | 19  |
|     |     | 23 |  |   |     |     |     | 29  | 31  |     |  |     | 37  |     |
| 41  |     | 43 |  |   |     | 47  |     |     |     | 53  |  |     |     | 59  |
| 61  |     |    |  |   |     | 67  |     |     | 71  | 73  |  |     |     | 79  |
|     |     | 83 |  |   |     |     |     | 89  |     |     |  |     | 97  |     |
| 101 | 103 |    |  |   | 107 | 109 |     |     |     | 113 |  |     |     |     |
|     |     |    |  |   | 127 |     |     | 131 |     |     |  | 137 | 139 |     |
|     |     |    |  |   |     | 149 |     | 151 |     |     |  | 157 |     |     |
|     | 163 |    |  |   | 167 |     |     |     | 173 |     |  |     |     | 179 |
| 181 |     |    |  |   |     |     |     | 191 | 193 |     |  | 197 | 199 |     |
|     |     |    |  |   |     |     |     | 211 |     |     |  |     |     |     |
|     | 223 |    |  |   | 227 | 229 |     |     | 233 |     |  |     |     | 239 |
| 241 |     |    |  |   |     |     |     | 251 |     |     |  | 257 |     |     |
|     | 263 |    |  |   |     | 269 |     | 271 |     |     |  | 277 |     |     |
| 281 | 283 |    |  |   |     |     |     |     |     | 293 |  |     |     |     |
|     |     |    |  |   | 307 |     |     | 311 | 313 |     |  | 317 |     |     |
|     |     |    |  |   |     |     |     | 331 |     |     |  | 337 |     |     |
|     |     |    |  |   | 347 | 349 |     |     | 353 |     |  |     |     | 359 |
|     |     |    |  |   | 367 |     |     |     | 373 |     |  |     |     | 379 |
|     | 383 |    |  |   |     |     | 389 |     |     |     |  | 397 |     |     |



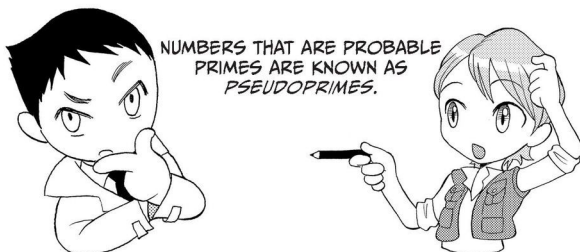




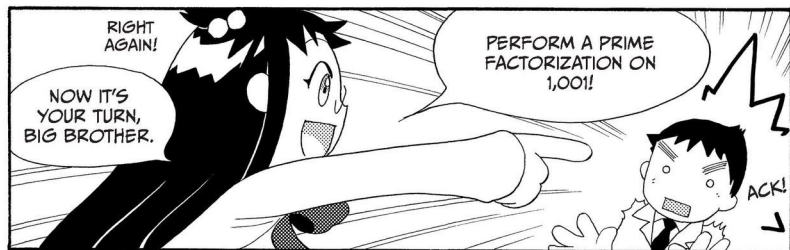
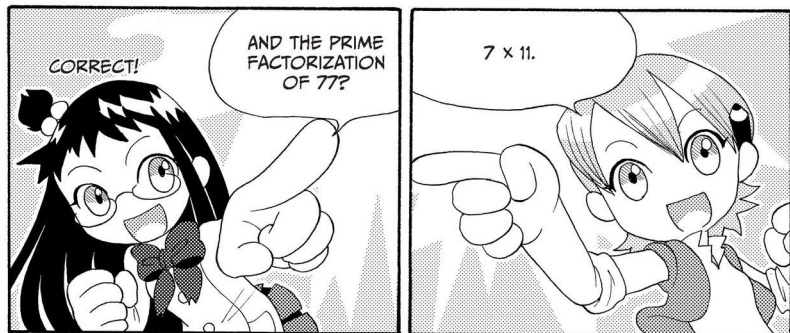
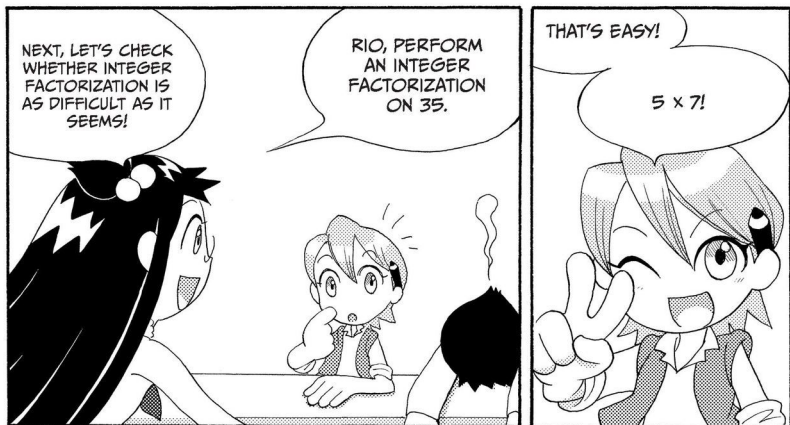
## TYPES OF PRIMALITY TESTS

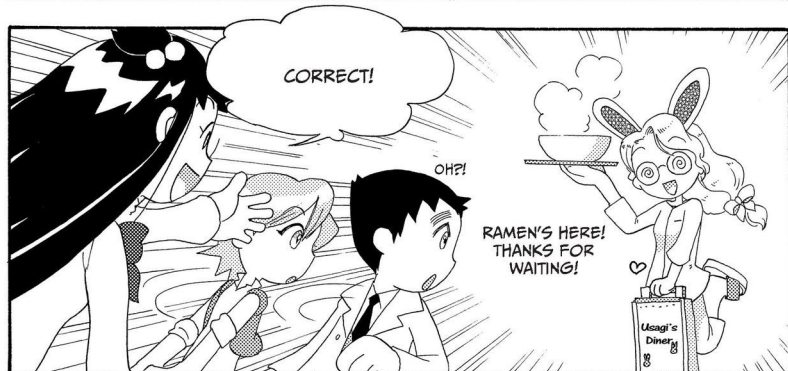
The sieve of Eratosthenes is a reliable method for finding prime numbers. When determining whether a very large number is prime, however, using this method takes a great deal of time. Instead, we can use Fermat's method. It isn't completely reliable because it determines prime numbers probabilistically, but it is faster. We'll cover Fermat's method in depth later, but for now, you just need to know that this method has a small chance of misidentifying a nonprime number  $n$  (a composite number) as prime.

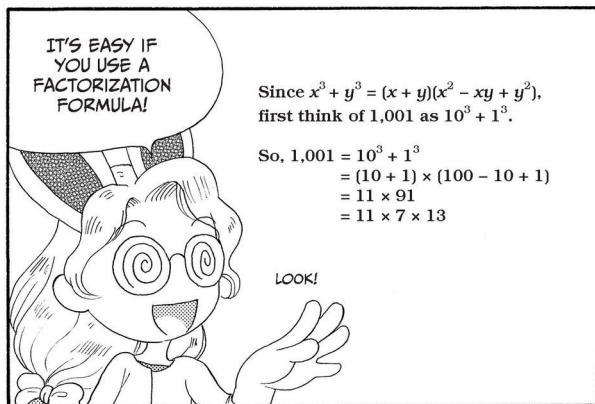
We could also use the Miller-Rabin method, which improves upon Fermat's method. In a given test, the probability of misidentifying a composite number as prime is less than with the Fermat method. In fact, this method can determine prime numbers with near certainty.

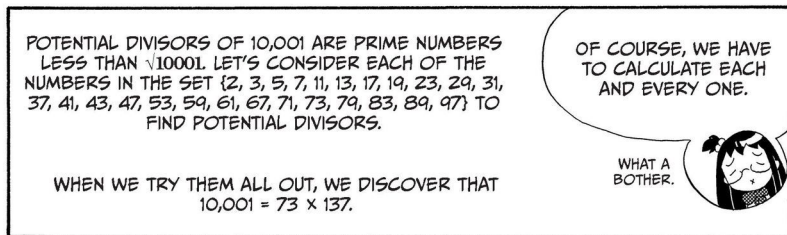
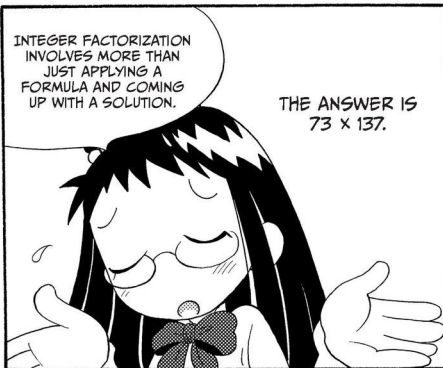














## MODULO OPERATIONS



NEXT, LET'S TAKE A LOOK  
AT THE REMAINDER OF  
INTEGER DIVISION.



YOU NEED TO BECOME  
FAMILIAR WITH MODULO  
OPERATIONS TO  
UNDERSTAND RSA  
ENCRYPTION.



YOU MEAN LIKE  
THE REMAINDER  
DIVISION YOU LEARN  
IN ELEMENTARY  
SCHOOL?

$15 \div 7 = 2$   
WITH A REMAINDER  
OF 1

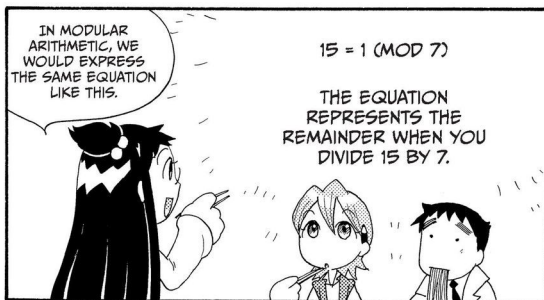


IN MODULAR  
ARITHMETIC, WE  
WOULD EXPRESS  
THE SAME EQUATION  
LIKE THIS.

$$15 = 1 \pmod{7}$$

THE EQUATION  
REPRESENTS THE  
REMAINDER WHEN YOU  
DIVIDE 15 BY 7.

WHAT DOES  
MOD MEAN?





IT'S AN ABBREVIATION  
FOR MODULO.

$$15 = 1 \pmod{7}$$

THIS EQUATION  
MEANS THAT WHEN 15 AND  
1 ARE DIVIDED BY 7, THEIR  
REMAINDER IS THE SAME.  
IN OTHER WORDS, 15 AND 1  
MODULO 7 ARE CONGRUENT.

In typical modular arithmetic,  $a = b \pmod{N}$  is also known as a modulo operation. This equation is read as "a and b are congruent modulo N." You can also use  $\equiv$  instead of  $=$  to note congruence.



mod

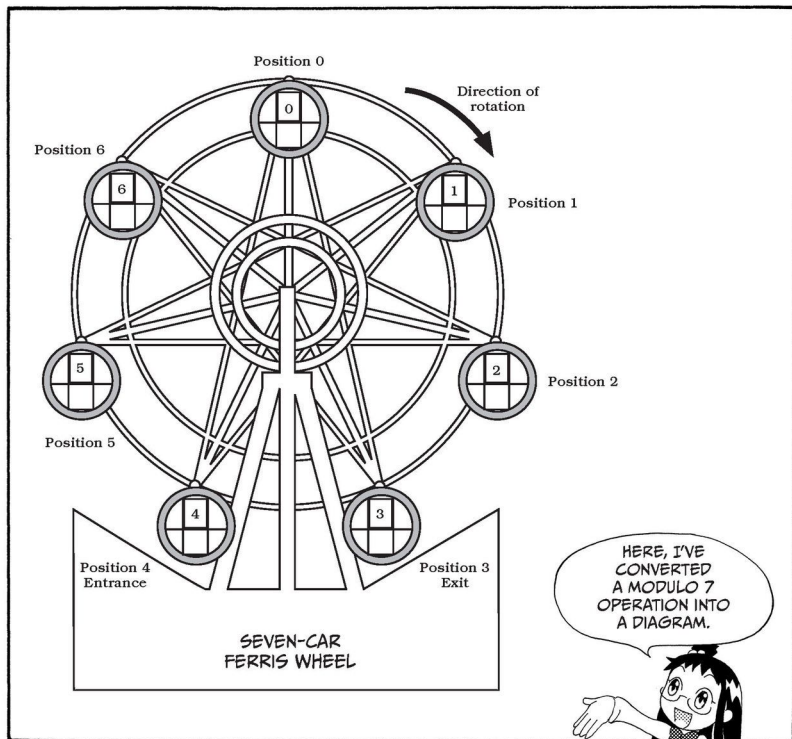
BUT IF IT'S JUST INTEGER  
CALCULATION, IT'S  
SIMPLE...

WHY COMPLICATE  
THINGS WITH MODULAR  
ARITHMETIC?

BECAUSE IT HAS A  
NUMBER OF VERY  
SPECIFIC ADVANTAGES  
FOR ENCRYPTION!

REALLY?

LIKE  
WHAT?



SINCE THE PROBLEM INVOLVES FINDING THE REMAINDER AFTER WE DIVIDE BY 7, WE'RE ONLY WORKING WITH THE NUMBERS 0 THROUGH 6.

THAT'S THE FIRST ADVANTAGE!

THE SURPLUS (REMAINDER) WILL ALWAYS BE SMALLER THAN THE DIVISOR  $N$  (MODULO  $N$  OR MOD  $N$ ).

SO OUR RESULTING VALUE IS LIMITED TO A SPECIFIC RANGE.

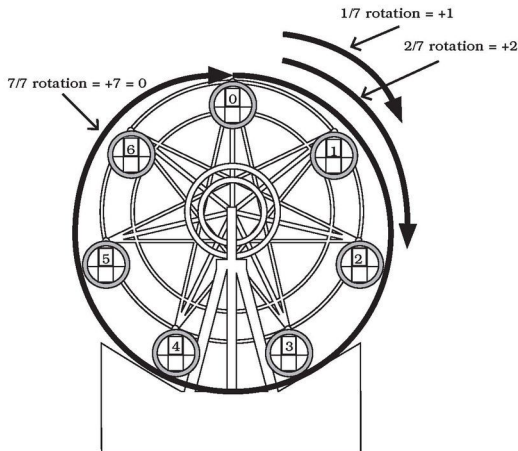


## ADDITION AND SUBTRACTION IN MODULO OPERATIONS

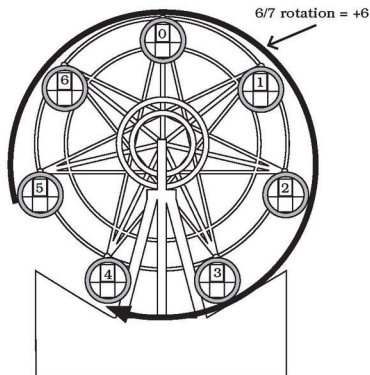
Let's use the Ferris wheel to model a modulo operation. Each of the seven cars is labeled with a number from 0 to 6. The cars correspond with numbered positions: we start with 0 at the top, and the positions are numbered 1 to 6 clockwise. Position 3 is the exit, and position 4 is the entrance.

Initially, car 0 is in position 0, car 1 is in position 1, and so forth. When we perform addition, the cars are rotated clockwise.

First, take a look at car 0. After moving  $1/7$  of a rotation, car 0 moves from position 0 to position 1. This is defined as  $+1$  (the addition of 1). When the wheel makes  $2/7$  of a rotation, car 0 moves from position 0 to position 2. This is  $+2$  (the addition of 2). When the wheel has made  $7/7$  of a rotation—in other words, one full rotation—car 0 returns once again to position 0. This is  $+7$ , which is the equivalent of 0—in other words, it is the same as if the wheel had not moved at all.



This Ferris wheel model can be used to explain modulo addition. For example, consider  $5 + 6$ . In the initial state, the 5 in  $5 + 6$  corresponds to car 5. When this car moves by  $6/7$  of a rotation, where do you think it will end up?



If we move the car six positions clockwise, we find that it ends up in position 4. In other words, we end up with the following equation:

$$5 + 6 = 4 \pmod{7}$$

When  $a$  is the car in its initial state (the car's initial position) and is moved  $b/7$  of a rotation, its final position is the solution to the addition problem. We can review all of the different additions here.

$a + b \text{ MODULO } 7$

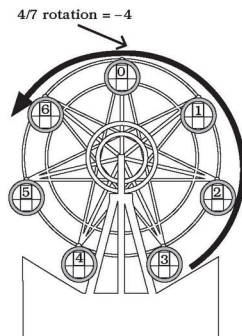
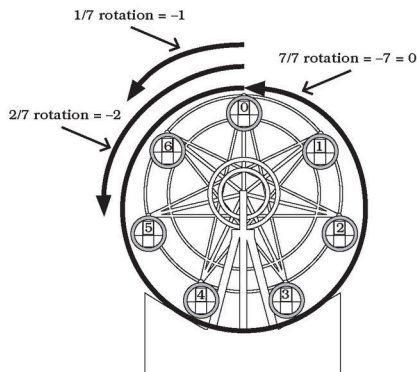
| $a \backslash b$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|------------------|---|---|---|---|---|---|---|
| 0                | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1                | 1 | 2 | 3 | 4 | 5 | 6 | 0 |
| 2                | 2 | 3 | 4 | 5 | 6 | 0 | 1 |
| 3                | 3 | 4 | 5 | 6 | 0 | 1 | 2 |
| 4                | 4 | 5 | 6 | 0 | 1 | 2 | 3 |
| 5                | 5 | 6 | 0 | 1 | 2 | 3 | 4 |
| 6                | 6 | 0 | 1 | 2 | 3 | 4 | 5 |

Next, let's use the Ferris wheel to explore subtraction.

First, take a look at car 0 in the left Ferris wheel as we put it through its paces. After moving counterclockwise by  $1/7$  of a rotation, car 0 moves from position 0 to position 6. This is defined as  $-1$  (a subtraction of 1). After moving counterclockwise by  $2/7$  of a rotation, car 0 moves from position 0 to position 5. This is  $-2$  (a subtraction of 2). If it moves a full rotation counterclockwise, car 0 returns to position 0. This is  $-7$ , which is also the equivalent of the car having not moved at all.

This Ferris wheel model can be used to explain all instances of subtraction. For example, let's consider the problem  $3 - 4$  using the Ferris wheel on the right.

In its initial state, the 3 in the expression  $3 - 4$  is car 3. After car 3 moves  $4/7$  of a rotation counterclockwise, it arrives at position 6.



In other words, this is the equivalent of the following equation:

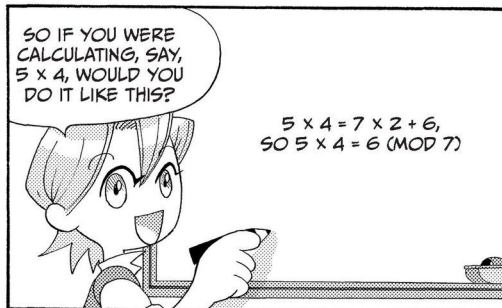
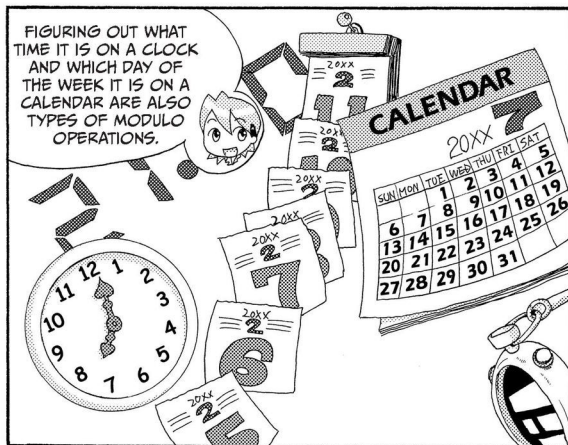
$$3 - 4 = 6 \pmod{7}$$

When  $a$  is the car in its initial state (the car's initial position) and is moved counterclockwise  $b/7$  of a rotation, its final position is the solution to our subtraction problem. The different combinations of subtracting  $a$  and  $b$  are shown in the table.

$a - b \text{ MODULO } 7$

| $a \backslash b$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|------------------|---|---|---|---|---|---|---|
| 0                | 0 | 6 | 5 | 4 | 3 | 2 | 1 |
| 1                | 1 | 0 | 6 | 5 | 4 | 3 | 2 |
| 2                | 2 | 1 | 0 | 6 | 5 | 4 | 3 |
| 3                | 3 | 2 | 1 | 0 | 6 | 5 | 4 |
| 4                | 4 | 3 | 2 | 1 | 0 | 6 | 5 |
| 5                | 5 | 4 | 3 | 2 | 1 | 0 | 6 |
| 6                | 6 | 5 | 4 | 3 | 2 | 1 | 0 |





$a \times b \text{ MODULO } 7$ 

| $\begin{smallmatrix} a \\ b \end{smallmatrix}$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|------------------------------------------------|---|---|---|---|---|---|---|
| 0                                              | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1                                              | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2                                              | 0 | 2 | 4 | 6 | 1 | 3 | 5 |
| 3                                              | 0 | 3 | 6 | 2 | 5 | 1 | 4 |
| 4                                              | 0 | 4 | 1 | 5 | 2 | 6 | 3 |
| 5                                              | 0 | 5 | 3 | 1 | 6 | 4 | 2 |
| 6                                              | 0 | 6 | 5 | 4 | 3 | 2 | 1 |

THE NUMBERS IN THIS  
TABLE SEEM PRETTY  
CHAOTIC, THOUGH...



BUT LOOK! IN THE  
COLUMNS IN WHICH  
NEITHER  $a$  NOR  $b$  IS  
EQUAL TO 0...

EACH NUMBER BETWEEN 1  
AND 6 APPEARS ONLY ONCE  
IN EACH COLUMN AND ROW!

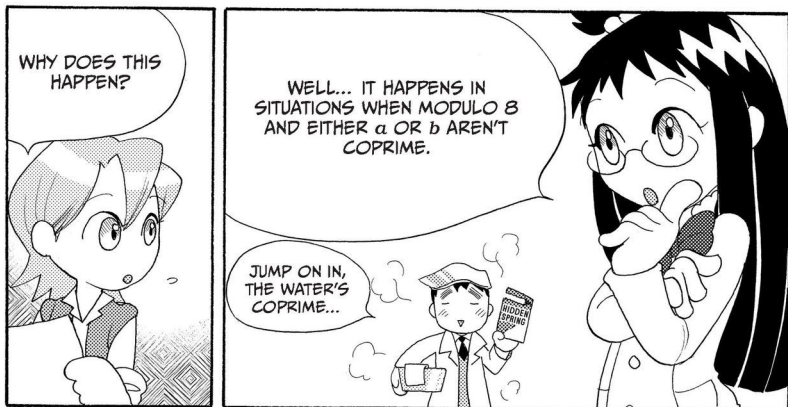
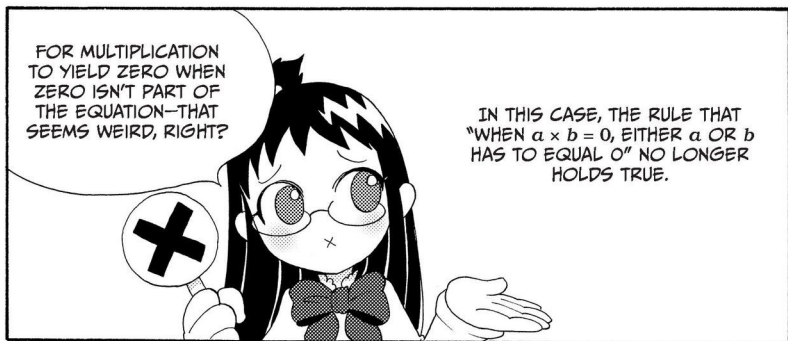
 $a \times b \text{ MODULO } 8$ 

| $\begin{smallmatrix} a \\ b \end{smallmatrix}$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------------------------------------------------|---|---|---|---|---|---|---|---|
| 0                                              | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1                                              | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2                                              | 0 | 2 | 4 | 6 | 0 | 2 | 4 | 6 |
| 3                                              | 0 | 3 | 6 | 1 | 4 | 7 | 2 | 5 |
| 4                                              | 0 | 4 | 0 | 4 | 0 | 4 | 0 | 4 |
| 5                                              | 0 | 5 | 2 | 7 | 4 | 1 | 6 | 3 |
| 6                                              | 0 | 6 | 4 | 2 | 0 | 6 | 4 | 2 |
| 7                                              | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

GREAT POINT!

HERE'S HOW  
MODULO 8  
MULTIPLICATION  
WORKS.



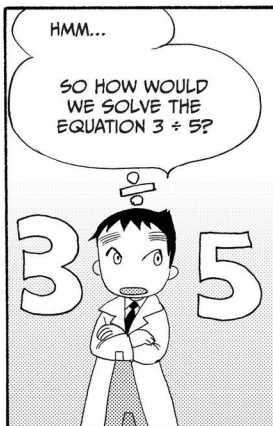




For example, in addition to 1, the numbers 8 and 2 share the common factor (common divisor) 2, so they aren't coprime. The numbers 4 and 6 in the modulo 8 multiplication table likewise share the common divisor 2 with 8, so they aren't coprime.

On the other hand, 1, 3, 5, and 7 are coprime to 8. You can determine whether two integers are coprime by confirming that the greatest common divisor of both is 1.

Thus, all prime numbers are coprime to integers that are not multiples of themselves. This property allows us to uncover prime numbers when using the sieve of Eratosthenes method.



JUST CONVERT  
THE DIVISION INTO  
MULTIPLICATION!

$$a \div b = a \times (1 / b)$$

DIVIDING  $a$  BY  $b$  IS THE  
SAME AS MULTIPLYING  $a$   
BY THE RECIPROCAL OF  $b$ .  
THE RECIPROCAL IS ALSO  
KNOWN AS THE INVERSE  
ELEMENT.

I SEE!

BUT WAIT,  
HOW DO YOU  
FIND THE  
RECIPROCAL?

$$3 \times (1/3) = 1$$

HUH?

FOR INSTANCE, THE  
RECIPROCAL OF 3 IS  $1/3$ .  
THE RECIPROCAL OF A  
NUMBER IS A NUMBER THAT  
YIELDS 1 WHEN MULTIPLIED  
BY THE ORIGINAL NUMBER.

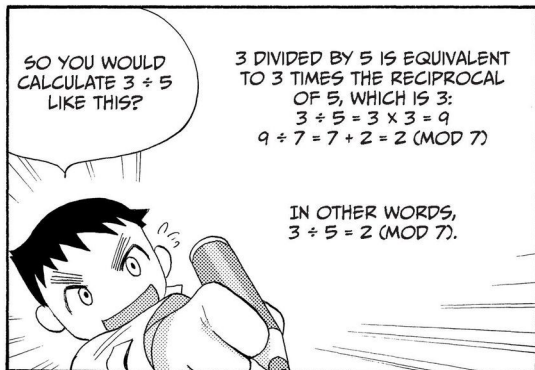
LET'S TAKE  
ANOTHER LOOK  
AT THE MODULO 7  
MULTIPLICATION  
TABLE AND FIND ALL  
INSTANCES OF 1.

THE RECIPROCAL OF 1 IS  
1, THE RECIPROCAL OF  
2 IS 4, THE RECIPROCAL  
OF 3 IS 5, THE  
RECIPROCAL OF 4 IS 2,  
THE RECIPROCAL OF 5 IS  
3, AND THE RECIPROCAL  
OF 6 IS 6.

$a \times b \text{ MODULO } 7$

| $a \backslash b$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|------------------|---|---|---|---|---|---|---|
| 0                | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1                | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2                | 0 | 2 | 4 | 6 | 1 | 3 | 5 |
| 3                | 0 | 3 | 6 | 2 | 5 | 1 | 4 |
| 4                | 0 | 4 | 1 | 5 | 2 | 6 | 3 |
| 5                | 0 | 5 | 3 | 1 | 6 | 4 | 2 |
| 6                | 0 | 6 | 5 | 4 | 3 | 2 | 1 |





THERE'S A DIVISION TABLE, TOO.

$a \div b \pmod{7}$

| $a \div b$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|------------|---|---|---|---|---|---|---|
| 0          | — | 0 | 0 | 0 | 0 | 0 | 0 |
| 1          | — | 1 | 4 | 5 | 2 | 3 | 6 |
| 2          | — | 2 | 1 | 3 | 4 | 6 | 5 |
| 3          | — | 3 | 5 | 1 | 6 | 2 | 4 |
| 4          | — | 4 | 2 | 6 | 1 | 5 | 3 |
| 5          | — | 5 | 6 | 4 | 3 | 1 | 2 |
| 6          | — | 6 | 3 | 2 | 5 | 4 | 1 |



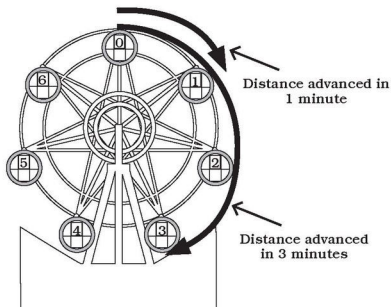
## MULTIPLICATION AND DIVISION OF MODULO OPERATIONS

Let's start by explaining multiplication using the seven-car Ferris wheel model. It's best to think of multiplication in terms of rotational speed.

In the initial state, car 0 is in position 0, car 1 is in position 1, and so on—all of the car numbers and position numbers match.

If in 1 minute, a car moves  $1/7$  of a rotation (in other words, in 7 minutes, it moves one full rotation), car 0's position after 3 minutes would be as follows:

$$1 \text{ (speed)} \times 3 \text{ (minutes)} = 3 \text{ (position after rotation)}$$



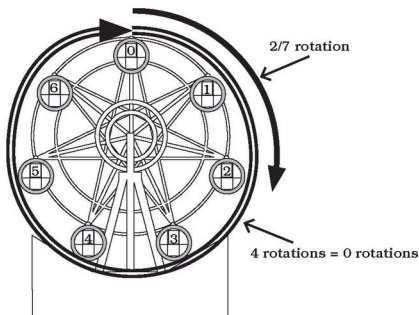
When the car moves at a rate of  $5/7$  of a rotation per minute, what position would it be in after 6 minutes?

$$5 \times 6 = 30$$

$$30 = 7 \times 4 + 2 \text{ (} 30 \div 7 = 4 \text{ with a remainder of 2)}$$

Therefore,  $5 \times 6 = 2 \pmod{7}$ .

In other words, after the value of 30 is divided by 7, we have 4 rotations with a remainder of 2. In this case, 4 rotations is equivalent to 0 rotations when dealing with modulo, so you only have to pay attention to the remainder of 2.



Think of division as the opposite of multiplication. Given the final position and the rotational speed, we determine the time the car was in motion using an inverse operation.

With a speed of  $1/7$  of a rotation per minute, car 0 begins at position 1 and finally ends up in position 5. Let's determine how many minutes the car was in motion.

$$5 \text{ (final position)} \div 1 \text{ (speed)} = 5 \text{ (minutes)}$$

So, it was in motion for 5 minutes. Were it to move for 12 minutes, or 19 minutes, it would end up in the same position, since each of these rotations is equal to 5 added to a multiple of 7 ( $5 + 7n$ ). However, because the times in our mod 7 calculation range only from 0 to 6 minutes, we can't visually represent the passage of 12 or 19 minutes. They would appear identical to 5 minutes. (See the Ferris wheel at the bottom left.)

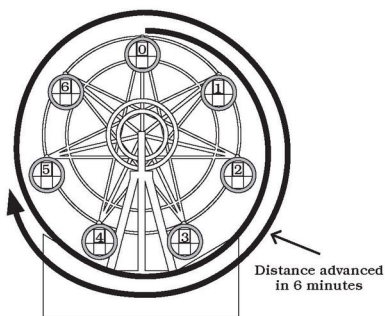
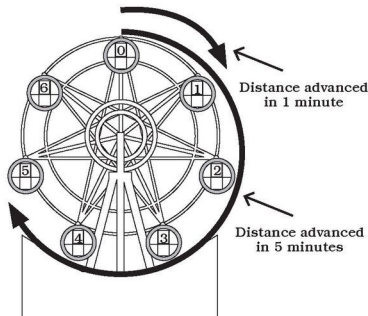
At a rate of  $2/7$  of a rotation per minute (in other words, two full rotations per 7 minutes), car 0 begins at position 0 and finally arrives at position 5. Using the division table, we can determine the number of minutes it was in motion as follows:

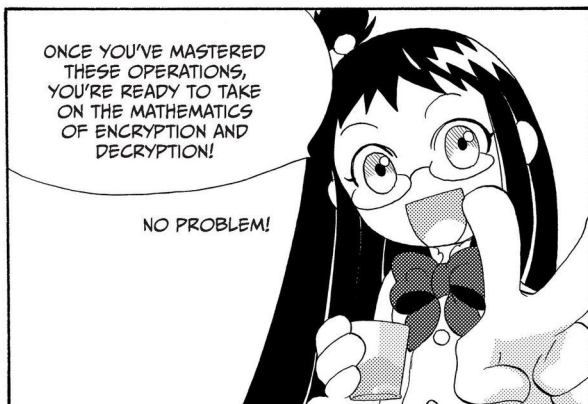
$$5 \text{ (final position)} \div 2 \text{ (speed)} = 6 \text{ (minutes)}$$

The answer is that it rotated for 6 minutes, but what's the best way to explain this?

Consider this: the final position was 5, but there was actually one surplus rotation. In other words, in reality the final position was  $2 \times 6 = 12$ , which in mod 7 is expressed as 5.

Therefore, since  $12 \div 2 = 6$ , the answer is that the car rotated for 6 minutes. (See the Ferris wheel at the right.)





MODULO OPERATIONS ON A PRIME NUMBER  $p$  ARE STILL SUBJECT TO THE COMMUTATIVE, ASSOCIATIVE, AND DISTRIBUTIVE PROPERTIES, SO ANY OPERATION WILL ALWAYS RESULT IN A NUMBER IN THE SET:  $\{0, 1, \dots, p-1\}$ . THIS SET IS EQUIVALENT TO THE CARS OF THE FERRIS WHEEL IN OUR EXAMPLE; WHEN THERE WERE 7 CARS, THEY WERE NUMBERED 0 THROUGH 6.

oof

EXAMPLES OF THE COMMUTATIVE PROPERTY:

$$a + b = b + a$$

$$ab = ba$$

EXAMPLES OF THE ASSOCIATIVE PROPERTY:

$$(a + b) + c = a + (b + c)$$

$$(ab)c = a(bc)$$

EXAMPLE OF THE DISTRIBUTIVE PROPERTY:

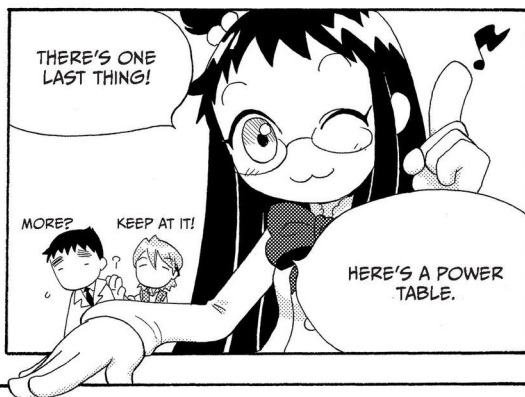
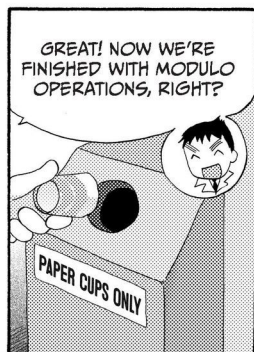
$$a(b + c) = ab + ac$$

A FIELD IS THE SET OF ELEMENTS (NUMBERS IN OUR CASE) RESULTING FROM MODULO AND ARITHMETIC OPERATIONS.\*

RATIONAL NUMBERS ARE ONE EXAMPLE OF A FIELD, SINCE ALL RATIONAL NUMBERS CAN BE FACTORS IN AN OPERATION. THERE IS AN INFINITE NUMBER OF RATIONAL NUMBERS, WHICH MAKES THE RATIONAL NUMBERS AN INFINITE FIELD. IN CONTRAST, THE FACTORS OF THE MODULO OPERATION FOR PRIME NUMBER  $p$  ARE  $0, 1, \dots, p-1$ , MEANING THAT  $p$  IS A FINITE FIELD BECAUSE THERE IS A LIMITED NUMBER OF NUMBERS IN THE FIELD.

\* THE JAPANESE WORD FOR FIELD SOUNDS SIMILAR TO THE WORD FOR A TYPE OF FISH.





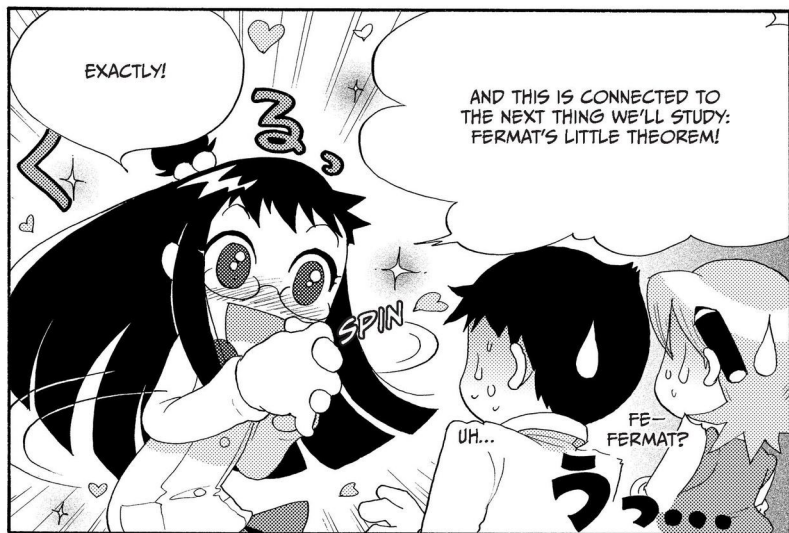
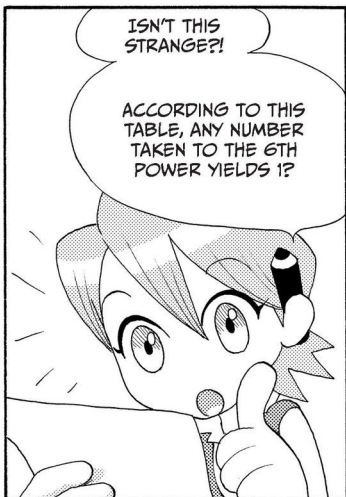
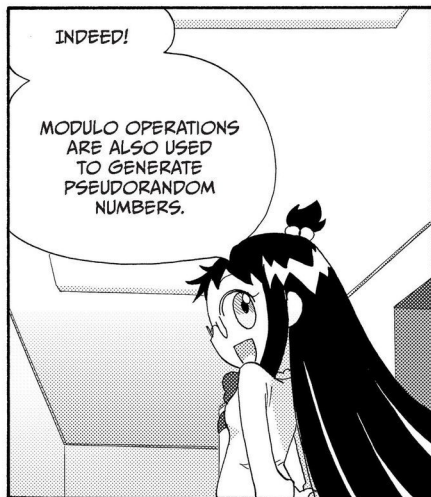
$a^b$  MODULO 7

| $a \backslash b$ | 1 | 2 | 3 | 4 | 5 | 6 |
|------------------|---|---|---|---|---|---|
| 1                | 1 | 1 | 1 | 1 | 1 | 1 |
| 2                | 2 | 4 | 1 | 2 | 4 | 1 |
| 3                | 3 | 2 | 6 | 4 | 5 | 1 |
| 4                | 4 | 2 | 1 | 4 | 2 | 1 |
| 5                | 5 | 4 | 6 | 2 | 3 | 1 |
| 6                | 6 | 1 | 6 | 1 | 6 | 1 |

TO CALCULATE THE VALUES IN THE TABLE, TAKE THE NUMBER  $a^b$  AND DIVIDE IT BY 7 TO GET THE REMAINDER. FOR EXAMPLE, TAKING ANY VALUE OF  $a$  BETWEEN 1 AND 6 TO THE 6TH POWER AND THEN DIVIDING THE RESULT BY 7 WILL ALWAYS RESULT IN A REMAINDER OF 1, WHICH IS REPRESENTED IN THE LAST COLUMN OF THE TABLE.

$$\begin{aligned}
 1^6 &= 1 = 0 \times 7 + 1 \\
 2^6 &= 64 = 9 \times 7 + 1 \\
 3^6 &= 729 = 104 \times 7 + 1 \\
 4^6 &= 4,096 = 585 \times 7 + 1 \\
 5^6 &= 15,625 = 2,232 \times 7 + 1 \\
 6^6 &= 46,656 = 6,665 \times 7 + 1
 \end{aligned}$$





## FERMAT'S LITTLE THEOREM AND EULER'S THEOREM



NOW,

LET'S MOVE ON TO  
FERMAT'S LITTLE  
THEOREM!

WE USED FERMAT'S LITTLE  
THEOREM IN PRIMALITY TESTS,  
REMEMBER?

IT'S AN ABSOLUTELY ESSENTIAL  
FOUNDATION TO HAVE FOR LEARNING  
EULER'S THEOREM.

FERMAT'S LITTLE THEOREM:

WHEN  $n$  IS A PRIME NUMBER  
AND YOU HAVE AN INTEGER  
 $a$  THAT IS COPRIME TO  $n$   
(AN INTEGER THAT IS NOT  
A MULTIPLE OF  $n$ ), THE  
FOLLOWING FORMULA  
HOLDS TRUE:

$$a^{n-1} = 1 \pmod{n}$$

IN OTHER WORDS, IF YOU  
TAKE  $a$  TO THE POWER OF  
 $n - 1$  AND DIVIDE IT BY  $n$ ,  
THE REMAINDER IS 1.

SO THIS IS THE LAW WE  
SAW DEMONSTRATED IN  
MODULO 7, WHEN RAISING  
NUMBERS 1 THROUGH 6 TO  
THE 6TH POWER YIELDS 1.

$a^b \text{ MODULO } 7$

| $a \backslash b$ | 1 | 2 | 3 | 4 | 5 | 6 |
|------------------|---|---|---|---|---|---|
| 1                | 1 | 1 | 1 | 1 | 1 | 1 |
| 2                | 2 | 4 | 1 | 2 | 4 | 1 |
| 3                | 3 | 2 | 6 | 4 | 5 | 1 |
| 4                | 4 | 2 | 1 | 4 | 2 | 1 |
| 5                | 5 | 4 | 6 | 2 | 3 | 1 |
| 6                | 6 | 1 | 6 | 1 | 6 | 1 |

WHO WAS THIS  
FERMAT GUY?



## FERMAT: THE FATHER OF NUMBER THEORY

Pierre de Fermat (1601–1665) was a 17th-century French lawyer and prominent mathematician. He made great contributions to the field of number theory, including but not limited to creating the modulo operation.

He created not only Fermat's little theorem but also what is known as Fermat's big theorem (or last theorem).

Fermat's big theorem states that given a natural number  $n$  whose value is 3 or more, no three natural numbers  $(x, y, z)$  can solve the equation  $x^n + y^n = z^n$ . Interestingly, Fermat himself left no proof of this theorem.

At first glance, the contents of the theorem look so simple that it seems like a middle schooler could solve it. I'm sure you're familiar with the Pythagorean theorem, which states that if the lengths of the sides of a right triangle are  $a$ ,  $b$ , and  $c$ , then  $a^2 + b^2 = c^2$ . Fermat's theorem concerns an equation for when the 2 in  $a^2 + b^2 = c^2$  is replaced by a value of 3 or more.

Proof of the big theorem came 330 years after Fermat's death, in 1995, thanks to British mathematician Andrew Wiles (1953–).

FERMAT HAD APPARENTLY  
WRITTEN THIS IN HIS  
NOTEBOOK:

I THOUGHT I WOULD WRITE A  
PROOF OF MY BIG THEOREM  
HERE, BUT THERE ISN'T ENOUGH  
SPACE IN THE MARGIN!

THERE, THERE.



LET'S USE FERMAT'S  
LITTLE THEOREM TO  
VERIFY PRIME NUMBERS!

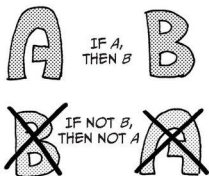


TAKE THE CONTRAPOSITION OF  
FERMAT'S LITTLE THEOREM. IF  $n$  AND  
 $a$  ARE COPRIME AND  $a^{(n-1)} \not\equiv 1 \pmod{n}$   
IS TRUE, THEN WE CAN SAY THAT  $n$  IS  
NOT A PRIME NUMBER.

WHAT'S  
CONTRAPOSITION?



A CONTRAPOSITION CONTRASTS  
WITH A PROPOSITION. FOR  
EXAMPLE, IF WE HAD THE  
PROPOSITION "IF  $A$ , THEN  $B$ ," THE  
CONTRAPOSITION WOULD BE "IF  
NOT  $B$ , THEN NOT  $A$ ."



THE CONTRAPOSITION  
OF A CORRECT  
PROPOSITION IS  
ALWAYS TRUE.

I SEE... WAIT,  
THIS ISN'T A  
MANGA.

QUIVER  
QUIVER

Fun  
Prime  
Numbers

IF THE PROPOSITION  
"ALL MANGA  
ARE FUN" IS  
TRUE, THEN THE  
CONTRAPOSITION  
"IF IT IS NOT FUN, IT  
IS NOT A MANGA" IS  
ALSO TRUE.

USING THIS AS A METHOD  
OF DETERMINING PRIME  
NUMBERS...

IS KNOWN AS  
FERMAT'S METHOD!

NO MORE...

OH DEAR



## FERMAT'S METHOD AND PSEUDOPRIME NUMBERS

To perform a primality test using Fermat's method, use this equation:

$$a^{n-1} \not\equiv 1 \pmod{n},$$

where  $n$ , the number you are performing the primality test on, and  $a$  are coprime.

Although Fermat's method is very efficient at testing for primality, even if the number  $n$  passes the primality test for several of its coprimes, that doesn't necessarily prove  $n$  is a prime number.

With Fermat's method, there are situations in which a number may be probabilistically determined to be prime even when it really isn't prime. We call such values *pseudoprime numbers*.

For example, when  $n = 3,215,031,751$ , it passes the primality test for the coprimes 2, 3, 5, and 7:

$$2^{3215031750} \equiv 1 \pmod{3,215,031,751}$$

$$3^{3215031750} \equiv 1 \pmod{3,215,031,751}$$

$$5^{3215031750} \equiv 1 \pmod{3,215,031,751}$$

$$7^{3215031750} \equiv 1 \pmod{3,215,031,751}$$

However, 3,215,031,751 is not actually a prime number. You can see this when you perform integer factorization:

$$3,215,031,751 = 151 \times 751 \times 28,351$$

However, out of the numbers  $n$  less than 250 billion, 3,215,031,751 is the only nonprime that passes the Fermat method's primality test for the four prime numbers 2, 3, 5, and 7.





## EULER'S THEOREM

Given a natural number  $n$  and a coprime integer  $a$ , we present the following equation:

$$a^{\phi(n)} = 1 \pmod{n}$$

The  $\phi(n)$  is known as *Euler's totient function*. Euler's totient function gives the number of integers between 1 and  $n$  that are coprime to  $n$ .

In addition, if we multiply  $a^{\phi(n)}$  by  $a$ , we get  $a^{\phi(n)+1}$ , and from this we can extrapolate the following equation:

$$a^{\phi(n)+1} = a \pmod{n}$$

When  $k$  is an integer,  $a^{k\phi(n)} \pmod{n}$  will result in 1, as we saw in this equation:

$$a^{\phi(n)} = 1 \pmod{n}$$

This might not be apparent at first sight, but for example, when  $k = 1$ ,

$$a^{k\phi(n)} = a^{1 \times \phi(n)} = a^{\phi(n)}$$

We can express an even number as  $k\phi(n) + 1$ , since adding 1 to any odd number will result in an even number. Because we know that multiplying both sides of Euler's theorem equation by  $a$  results in

$$a^{\phi(n)+1} = a \pmod{n},$$

that means modding  $ak^{\phi(n)+1}$  by  $n$  will also result in  $a$ . You should get the following formulas:

$$ak^{\phi(n)} = 1 \pmod{n}$$

$$ak^{\phi(n)+1} = a \pmod{n}$$

This means that when  $a$  is any integer between 1 and  $(n - 1)$ , the following formula will apply:

$$a^{k\phi(n)+1} = a \pmod{n}$$

GIVEN  $\phi(7)$ , THE NUMBERS  
 $\{1, 2, 3, 4, 5, 6\}$  ARE  
COPRIME TO 7.

SO  $\phi(7) = 6$ , RIGHT?

IF  $n$  IS A PRIME NUMBER,  
THEN THE ONLY NUMBER  
THAT ISN'T COPRIME TO  
 $n$  BETWEEN THE NATURAL  
NUMBERS 1 AND  $n$  IS  $n$   
ITSELF!

IN OTHER WORDS,  
 $\phi(n) = n - 1$ .

WHEN  $n$  IS A PRIME  
NUMBER, THEN  
 $\phi(n) = (n - 1)$ , WHICH WE  
CAN SUBSTITUTE INTO THE  
EQUATION SO THAT  
 $a^{\phi(n)} = a^{n-1} = 1 \pmod{n}$ ,  
WHICH IS CONSISTENT  
WITH FERMAT'S LITTLE  
THEOREM AND THE GRAY  
COLUMN OF THE TABLE  
ON PAGE 152.

## THE MATHEMATICIAN EULER

Leonhard Euler (1707–1783) was a prominent Swiss mathematician of the 18th century.

He not only made huge contributions to the broad field of mathematics but also contributed to the fields of physics and astronomy. His most well-known contribution to mathematics is known as *Euler's formula*:

$$e^{i\theta} = \cos\theta + i(\sin\theta)$$

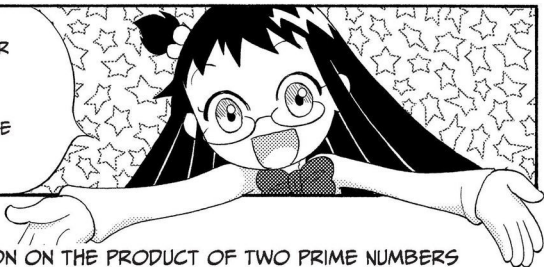
We won't cover it in detail, but this formula demonstrates the relationship between the complex exponential function  $e^{i\theta}$  and the trigonometric functions  $\cos\theta$  and  $\sin\theta$  through the imaginary number  $i = \sqrt{-1}$ .



WHAT KIND OF  
GUY WAS THIS  
EULER?

I LOVE STARS,  
TOO.

LET'S LOOK AT AN EULER FUNCTION IN WHICH A NONPRIME NUMBER  $N$  IS EXPRESSED AS THE PRODUCT OF TWO PRIME NUMBERS  $p$  AND  $q$ .



## EULER FUNCTION ON THE PRODUCT OF TWO PRIME NUMBERS

$N$  is the product of two prime numbers  $p$  and  $q$ . Here, in order to derive the Euler function of  $N$  using  $p$  and  $q$ , we need to first count the integers that are not coprime to  $N$ . Since  $p$  and  $q$  are prime numbers, we know that multiples of  $p$  and multiples of  $q$  make up all the numbers that are not coprime to  $N$ .

1. Because the multiples of  $p$  between 1 and  $qp$  are  $p, 2p, 3p, \dots$ , up to  $qp$ , there are  $q$  number of multiples of  $p$ .
2. Because the multiples of  $q$  between 1 and  $qp$  are  $q, 2q, 3q, \dots$ , up to  $pq$ , there are  $p$  number of multiples of  $q$ .
3. Of the multiples of  $q$  and  $p$ , only  $qp$  and  $pq$  are shared by both  $p$  and  $q$ , but because  $qp$  and  $pq$  are both equal to  $N$ , they are the same number.

For a prime number  $n$ ,  $\phi(n) = n - 1$ , but because  $N$  isn't prime and the multiples of the prime numbers  $q$  and  $p$  are not coprime with it, we need to subtract the number of multiples of  $p$  and  $q$  from  $N$  instead. Through our analysis from steps 1 and 2, we found that there are  $q + p$  number of integers not coprime with  $N$ . In other words,

$$\phi(N) = N - p - q + 1$$

Since  $N = pq$ , we can substitute it into the equation to get this:

$$\phi(N) = pq - p - q + 1 = (p - 1)(q - 1)$$

This means that  $\phi(pq) = \phi(p)\phi(q)$ . If we apply Euler's theorem to  $p$  and  $q$  individually, we would get these equations:

$$a^{p-1} = 1 \pmod{p}$$

$$a^{q-1} = 1 \pmod{q}$$

We can combine these equations to derive another equation. The smallest multiple  $(p - 1)$  and  $(q - 1)$  share is known as their *lowest common multiple*, which we'll represent as  $L$ . This means that when  $(p - 1)$  is multiplied by some integer  $m$ , it equals  $L$ , and when  $(q - 1)$  is multiplied by some integer  $n$ , it also equals  $L$ . Therefore,  $L = m(p - 1) = n(q - 1)$ .

With  $a^{p-1} = 1 \pmod{p}$ , if we raise both sides by  $m$ , we get

$$a^{(p-1)m} = 1^m \pmod{p}$$

$$a^L = 1 \pmod{p}$$

If we do the same with  $\alpha^{q-1} = 1 \pmod{q}$ , we get the same result:

$$\alpha^{(q-1)n} = 1^n \pmod{q}$$

$$\alpha^L = 1 \pmod{q}$$

We can combine these two equations to get  $\alpha^L = 1 \pmod{p, \text{ mod } q}$ . Because  $N$  is equal to  $pq$ , we would get the following equation:

$$\alpha^L = 1 \pmod{N}$$

In other words,  $L$  functions identically to Euler's theorem  $\phi(N)$ . In addition, because the product of two arbitrary positive integers is equal to the product of the lowest common multiple and the greatest common divisor (which we'll call  $G$ ), we would get the equation  $(p-1)(q-1) = LG$ , and we can derive the following:

$$L = \frac{(p-1)(q-1)}{G}$$

For example, let's say  $p = 3$  and  $q = 5$ , so  $N = pq$  is 15. Therefore,  $(p-1)$  is 2,  $(q-1)$  is 4, and  $\phi(N) = (p-1)(q-1)$  is 8. The lowest common multiple  $L$  of 2 and 4 is 4, and the greatest common divisor  $G$  is 2. In this case, for any natural number  $a$  that is coprime to 15 and for any  $k$  that is a nonnegative integer, the following formula applies:

$$\alpha^{4k} = 1 \pmod{15}$$

In other words, until we take  $a$  to the  $\phi(N)$ th power, the greatest common divisor  $G = 1$  sometimes appears periodically for every  $L$  numbers. According to the Euler's theorem equation, for all of the integers  $a$  between 1 and  $(N-1)$ , this applies:

$$\alpha^{kL+1} = a \pmod{N}$$

You can see this in the gray columns of this table.

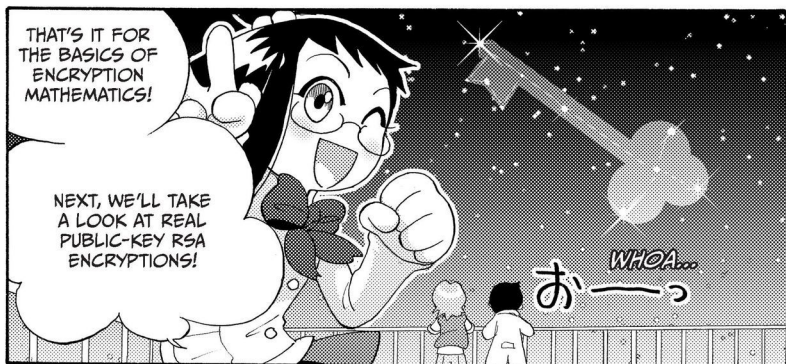
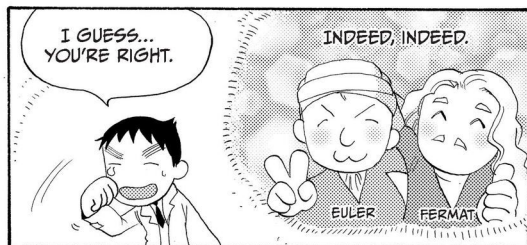
PRODUCT OF TWO PRIME NUMBERS WHEN  $\alpha^b$  ( $N = 3 \times 5$ ,  $\phi(15) = 8$ ,  $L = 4$ ,  $G = 2$ )

| $\alpha \backslash b$ | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  |
|-----------------------|----|----|----|----|----|----|----|----|
| 1                     | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 2                     | 2  | 4  | 8  | 1  | 2  | 4  | 8  | 1  |
| 3                     | 3  | 9  | 12 | 6  | 3  | 9  | 12 | 6  |
| 4                     | 4  | 1  | 4  | 1  | 4  | 1  | 4  | 1  |
| 5                     | 5  | 10 | 5  | 10 | 5  | 10 | 5  | 10 |
| 6                     | 6  | 6  | 6  | 6  | 6  | 6  | 6  | 6  |
| 7                     | 7  | 4  | 13 | 1  | 7  | 4  | 13 | 1  |
| 8                     | 8  | 4  | 2  | 1  | 8  | 4  | 2  | 1  |
| 9                     | 9  | 6  | 9  | 6  | 9  | 6  | 9  | 6  |
| 10                    | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 11                    | 11 | 1  | 11 | 1  | 11 | 1  | 11 | 1  |
| 12                    | 12 | 9  | 3  | 6  | 12 | 9  | 3  | 6  |
| 13                    | 13 | 4  | 7  | 1  | 13 | 4  | 7  | 1  |
| 14                    | 14 | 1  | 14 | 1  | 14 | 1  | 14 | 1  |

1 cycle

1 cycle





# THE STRUCTURE OF RSA ENCRYPTION



LIBERATED FROM  
MATH AT LAST!

SWEET,  
SWEET  
SUGAR

WHAT DO YOU  
MEAN? WE'RE  
JUST GETTING  
STARTED!

THESE SNACKS ARE  
HERE TO TEACH YOU  
ABOUT ENCRYPTION.

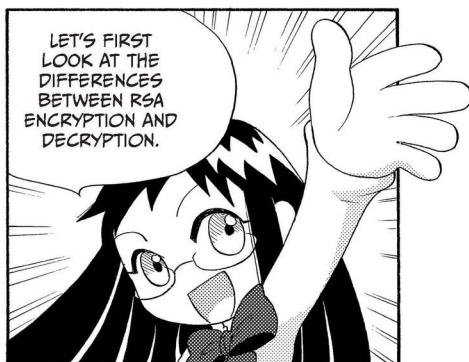
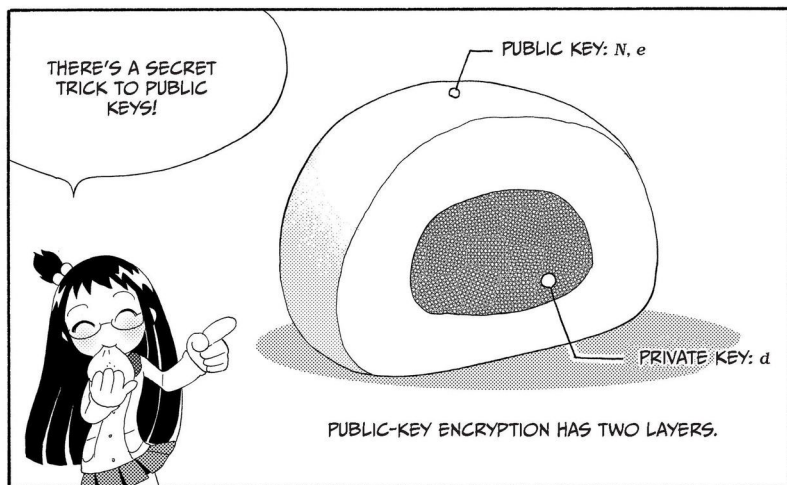
I CAN  
FINALLY  
RELAX.

WHAAAT!

THE SECRET TO  
RSA ENCRYPTION  
KEYS...

IS ACTUALLY THIS!

HUH?  
A PORK BUN?



## RSA ENCRYPTION: ENCRYPTION AND DECRYPTION

With a plaintext message  $m$  and ciphertext  $C$ , encryption is expressed as follows:

$$C = m^e \pmod{N}$$

In other words, if we raise  $m$  (the plaintext) to the power of  $e$  (a public key) and divide it by  $N$  (another public key), we're left with  $C$  (the ciphertext).

Decryption is expressed as follows:

$$m = C^d \pmod{N}$$

In other words, in decryption,  $C$  raised to the power of  $d$  (the private key) divided by  $N$  yields  $m$ . Here,  $N$  is made up of two different and large prime numbers  $p$  and  $q$ .



DO YOU KNOW WHY WE CAN'T DECODE THE FIRST EQUATION, EVEN THOUGH THE PUBLIC KEYS  $e$  AND  $N$  AND THE CIPHERTEXT  $C$  ARE KNOWN?



IF YOU THINK OF  $m$  IN TERMS OF AN UNKNOWN VALUE  $x$  IN AN EQUATION...

I DON'T WANNA HEAR IT!



IF YOU KNOW EVERYTHING  
BUT  $x$  (WHICH IS  $m$ ) AND  
 $d$ , YOU THINK IT'D BE  
POSSIBLE TO DECIPHER  
THE CIPHERTEXT.

RIGHT?

BUT SINCE IT WOULD TAKE AN  
ENORMOUS AMOUNT OF TIME  
TO DERIVE  $x$  BY SUBSTITUTING  
NUMERICAL VALUES FOR  $d$  ONE  
BY ONE,

IT WOULD BE NEARLY IMPOSSIBLE  
TO SOLVE THE EQUATION FROM A  
PRACTICAL STANDPOINT. THIS IS WHAT  
MAKES RSA A COMPUTATIONALLY  
SECURE ENCRYPTION.

IF YOU UNDERSTAND  
EULER'S TOTIENT  
FUNCTION  $\phi(N)$ , YOU  
CAN CALCULATE THE  
RESULT USING EULER'S  
THEOREM!

BUT WHEN  
CALCULATING  
 $\phi(N)$ , YOU HAVE TO  
PERFORM INTEGER  
FACTORIZATION ON  
 $N$ , RIGHT?

SQUEEZE

IF  $N$  IS AN  
INCREDIBLY LARGE  
NUMBER, INTEGER  
FACTORIZATION WILL  
TAKE AN ENORMOUS  
AMOUNT OF TIME,  
WHICH MEANS THE  
ENCRYPTION'S  
SECURITY IS BASED  
ON INTEGER  
FACTORIZATION.

LISTEN  
CLOSELY!

IN OTHER WORDS, WE END UP WITH AN INTEGER  
FACTORIZATION PROBLEM THAT IS DIFFICULT TO  
SOLVE MATHEMATICALLY!

THAT'S WHY IT'S HARD TO DECRYPT!

THE ENCRYPTION KEY  
AND DECRYPTION KEY  
CARRY OUT IMPORTANT  
ROLES IN A PUBLIC KEY.  
NEXT UP, LET'S LEARN  
ABOUT THE METHODS  
FOR GENERATING THEM  
STEP-BY-STEP!

IS HE  
OKAY?



## GENERATING AN RSA ENCRYPTION KEY

We'll first generate the public key, and then we'll generate the private key.

### PUBLIC-KEY GENERATION

Public-key generation involves several steps.

1. Select two different, large prime numbers,  $p$  and  $q$ , at random and calculate  $N = pq$ .

2. Solve Euler's function  $\phi(N) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$ .

After you solve Euler's function,  $p$  and  $q$  become unnecessary! You should discard them so that other parties don't find them. This is crucial because if an adversary finds  $p$  and  $q$ , it is trivial to then calculate  $d$  and break RSA encryption.

3. Select a random positive integer  $e$  such that  $e$  is coprime with  $N$ :  
 $1 < e < \phi(n)$

However, select  $e$  such that  $m^e > N$ ! If  $m^e \leq N$ , we wouldn't be able to use the modulo operation since we would get  $m^e = C$ , where  $C$  is the ciphertext. We wouldn't be able to mod by  $N$  to scramble the equation. Because we don't know what messages we'll be encrypting, it is best to choose a reasonably large  $e$  (in the past, 65,537 has commonly been used as a value for  $e$ ).

### PRIVATE-KEY GENERATION

Find a positive integer  $d$  that satisfies the following equations:

$$d = e^{-1} \pmod{\phi(N)}$$

$$ed = 1 \pmod{\phi(N)},$$

where  $d$  is less than  $\phi(N)$  but larger than both  $p$  and  $q$ .

Now the public key is the pair  $(n, e)$ , and the private key is  $d$ .



### VERIFYING THAT THE PUBLIC KEY $E$ AND PRIVATE KEY $D$ WORK IN RSA ENCRYPTION

Since we know that  $ed = 1 \pmod{\phi(n)}$ , we can subtract 1 from each side of the equation to get  $ed - 1 = 0 \pmod{\phi(n)}$ . In other words,  $ed - 1$  leaves no remainder when modded by  $\phi(n)$ , meaning that it is a multiple of  $\phi(n)$ . So, you get

$$ed - 1 = k \phi(n),$$

where  $k$  is a nonnegative integer. In addition, we can manipulate the equation to be this:

$$ed = k \phi(n) + 1$$

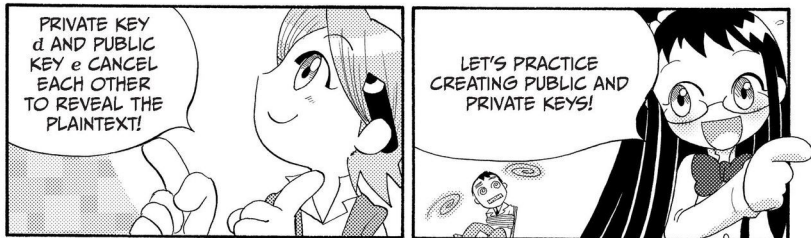
Therefore, if we take the earlier equation  $a^{kl+1} = a \pmod{N}$  (see page 159), we can express all of the natural numbers of  $m$  (the plaintext) from 1 to  $(N - 1)$  in the following equation:

$$m^{kl+1} = m \pmod{N}$$

Because  $ed = k\phi(n) + 1$ , we can substitute the exponent, so when the ciphertext is raised to the private key, we get the following:

$$m^{ed} = m^{k\phi(n)+1} = m \pmod{N}$$

Since the ciphertext  $C = m^e$ ,  $m^{ed} = C^d$ , it becomes clear that  $d$  allows us to decrypt plaintext  $m$ .



## HOW TO MAKE PUBLIC KEYS AND PRIVATE KEYS

Now, given two prime numbers  $p = 5$  and  $q = 11$ , let's solve for public key  $N$  and private key  $d$ .

### STEP 1

The product of  $p$  and  $q$  is  $N$ .

$$N = pq = 5 \times 11 = 55$$

### STEP 2

Solve the Euler function of  $N$ ,  $\phi(N)$ .

$$\phi(55) = (5 - 1) \times (11 - 1) = 4 \times 10 = 40$$

### STEP 3

Choose a random integer  $e$  such that  $e$  is coprime to  $N$  and  $1 < e < \phi(N)$ . The list of candidates that fulfill these requirements for our current example is  $\{1, 3, 7, 9, 11, 13, 17, 19\}$ . We'll use  $e = 17$ , so the public key is  $(55, 17)$ .

### STEP 4

Solve for the private key  $d$ . We know that  $e = 17$ , and we are using modulo  $\phi(N)(40)$ . The equation we need to solve is

$$d = e^{-1} \pmod{\phi(N)}$$

$$ed = 1 \pmod{\phi(N)}$$

$$17d = 1 \pmod{40}$$

We can see that  $d$  multiplied by 17 modulo 40 must equal 1. From the extended Euclidean algorithm, which we'll see later, we get that  $d = 33$ .

For now, let's verify that:

$$17d = 1 \pmod{40}$$

$$17 \times 33 = 1 \pmod{40}$$

$$561 = 1 \pmod{40}$$

where

$$561 = 1 + 40k$$

$$561 = 1 + (40 \times 14) \text{ for } k = 14$$

$$561 = 1 + 560$$

This shows that  $d = 33$  is valid; thus the private key is  $d = 33$ .

# STEP 5

Solve for the decryption key  $d$ , which is the inverse of encryption key  $e$ . We know that  $e = 17$  and are modding by 40 ( $\phi(55) = 40$ ).

$$ed = k\phi(n) + 1 \pmod{40}, \text{ so } 17d = 40k + 1$$

When you rearrange this equation, you get

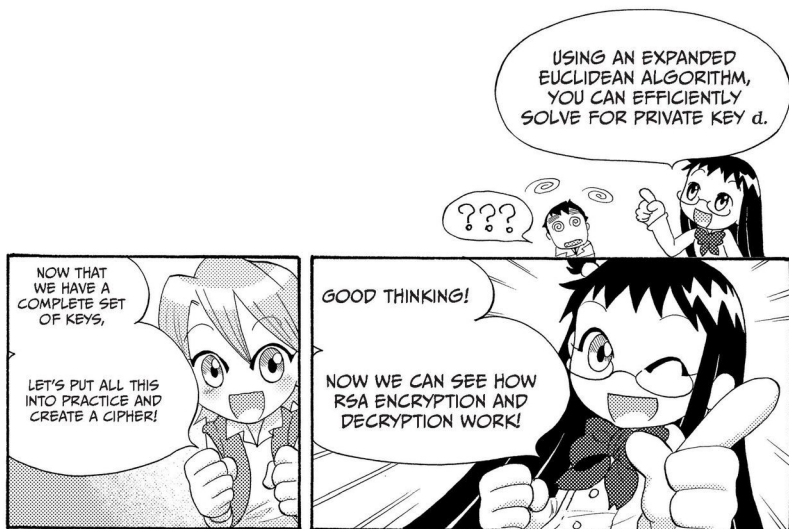
$$\frac{40k + 1}{17}$$

Because we know  $d$  must be an integer, the right side of the equation can't leave a remainder. So we can search for solutions in which  $40k + 1$  is a multiple of 17 and find that one solution is  $k = 11$ .

Furthermore, you can calculate what  $d$  is in relation to each  $e$  acquired in step 4 by solving the same formula,  $ed = 40k + 1$ . You would get the following  $d$  and  $e$  pairs:

$$(e = 3, d = 27), (e = 7, d = 23), (e = 9, d = 9), \\ (e = 11, d = 11), (e = 17, d = 33), (e = 19, d = 19)$$

In general, the encryption key and decryption key should be different integers ( $e \neq d$ ), and larger integers are preferred for encryption key  $e$ ;  $e = 17$  and  $d = 33$  are just examples we will use here.



## GENERATING A CIPHERTEXT USING RSA

To begin, we'll look at the procedure for using a public key to generate a ciphertext.

To make this a concrete example, we'll encrypt the four-letter plaintext GOLF using the encryption key  $e = 17$  that we calculated earlier.

### STEP 1

First, use the table at the right to encode the characters as integers. This is non-standard character encoding, but it is convenient for this exercise.

|    |    |    |    |
|----|----|----|----|
| G  | O  | L  | F  |
| ↓  | ↓  | ↓  | ↓  |
| 32 | 40 | 37 | 31 |

### STEP 2

Convert the integers to 6-bit binary.

|        |        |        |        |
|--------|--------|--------|--------|
| 32     | 40     | 37     | 31     |
| ↓      | ↓      | ↓      | ↓      |
| 100000 | 101000 | 100101 | 011111 |

### STEP 3

Express the binary data as a nonnegative integer less than  $(N - 1)$ . In this example, since  $N = 55$  and  $N - 1 = 55 - 1 = 54$ , the data is divided into 5-bit units. In other words, the greatest value that can be expressed in 5 bits is 31, which is below 54 and fulfills the conditions. Of course, we could use 3- or 4-bit sections instead, but using larger-bit sections is more efficient.

|           |           |           |           |         |
|-----------|-----------|-----------|-----------|---------|
| 100000    | 101000    | 100101    | 011111    | 0 ← Add |
| ↓ ↓ ↓ ↓ ↓ | ↓ ↓ ↓ ↓ ↓ | ↓ ↓ ↓ ↓ ↓ | ↓ ↓ ↓ ↓ ↓ | ↓       |
| 10000     | 01010     | 00100     | 10101     | 11110   |

We append the final 0 to pad the last section because there is an insufficient number of bits to make the section a complete 5-bit unit. Here, we're using 0, but other padding digits could be used instead, depending on the encryption scheme.

### STEP 4

Convert the binary data into decimal notation.

|       |       |       |       |       |
|-------|-------|-------|-------|-------|
| 10000 | 01010 | 00100 | 10101 | 11110 |
| ↓     | ↓     | ↓     | ↓     | ↓     |
| 16    | 10    | 4     | 21    | 30    |

### CHARACTER ENCODING

| Character | Code | Character | Code |
|-----------|------|-----------|------|
| a         | 0    | K         | 36   |
| b         | 1    | L         | 37   |
| c         | 2    | M         | 38   |
| ...       | ...  | N         | 39   |
| w         | 22   | O         | 40   |
| x         | 23   | P         | 41   |
| y         | 24   | Q         | 42   |
| z         | 25   | R         | 43   |
| A         | 26   | S         | 44   |
| B         | 27   | T         | 45   |
| C         | 28   | U         | 46   |
| D         | 29   | V         | 47   |
| E         | 30   | W         | 48   |
| F         | 31   | X         | 49   |
| G         | 32   | Y         | 50   |
| H         | 33   | Z         | 51   |
| I         | 34   | ...       | ...  |
| J         | 35   | Space     | 63   |



STEP 5

Using the encryption keys ( $N = 55$ ,  $e = 17$ ), perform the encryption. In concrete terms, this involves raising the data in decimal notation to the 17th power, dividing by 55, and searching for the remainder. By performing these calculations for each integer as follows . . .

$$16^{17} \pmod{55}, 10^{17} \pmod{55}, 4^{17} \pmod{55}, 21^{17} \pmod{55}, 30^{17} \pmod{55}$$

. . . we obtain the encrypted data. For example, we can calculate this for 16 by finding the solution to  $16^{17} \pmod{55}$ . We can reduce  $16^{17}$  by expanding the exponent as follows:

$$16^{17} = 16^2 \times 16^2 \times 16^2 \times 16^2 \times 16^2 \times 16^2 \times 16^2 \times 16 \pmod{55}$$

Because  $16^2 = 256 = 36 \pmod{55}$ , we can replace each  $16^2$  with 36:

$$= 36 \times 36 \times 36 \times 36 \times 36 \times 36 \times 36 \times 16 \pmod{55}$$

which is equal to

$$= 36^2 \times 36^2 \times 36^2 \times 36^2 \times 16 \pmod{55}$$

And since  $36^2 = 1,296 = 31 \pmod{55}$ , we can further reduce the equation:

$$= 31 \times 31 \times 31 \times 31 \times 16 \pmod{55}$$

$$= 31^2 \times 31^2 \times 16 \pmod{55}$$

Then because  $31^2 = 961 = 26 \pmod{55}$ , we get

$$= 26 \times 26 \times 16 \pmod{55}$$

$$= 26^2 \times 16 \pmod{55}$$

And finally, because  $26^2 = 676 = 16 \pmod{55}$ , we get

$$= 16 \times 16 \pmod{55}$$

$$= 36 \pmod{55}$$

The remaining encryption is performed in the same way, and we record the results as follows:

$$10^{17} \pmod{55} = 10$$

$$4^{17} \pmod{55} = 49$$

$$21^{17} \pmod{55} = 21$$

$$30^{17} \pmod{55} = 35$$

Thus, the ciphertext expressed in decimal notation is as follows:

$$36 \ 10 \ 49 \ 21 \ 35$$

Then we can encode the decimal notation into letters:

|    |    |    |    |    |
|----|----|----|----|----|
| 36 | 10 | 49 | 21 | 35 |
| ↓  | ↓  | ↓  | ↓  | ↓  |
| K  | k  | X  | v  | J  |

So the ciphertext is KkXvJ.

## DECRYPTING A CIPHERTEXT USING RSA

Now let's go over the process of using the private key to decrypt a ciphertext into plaintext. We'll use the private encryption key  $d = 33$  in our concrete example.

### STEP 1

Using decryption key  $d = 33$ , calculate  $C^d \pmod{N}$ . This involves raising each number in the decimal-form ciphertext (36 10 49 21 35) to the 33rd power, dividing by 55, and seeking out the remainder to obtain the plaintext data as follows:

$$36^{33} \pmod{55}, 10^{33} \pmod{55}, 49^{33} \pmod{55}, 21^{33} \pmod{55}, 35^{33} \pmod{55}$$

To calculate this for the first decimal digit 36, adopt the same technique used in step 5 of generating the ciphertext, as follows:

$$\begin{aligned} 36^{33} &= 36^3 \times 36^3 \times 36^3 \times 36^3 \times 36^3 \times 36^3 \times 36^3 \times 36^3 \times 36^3 \times 36^3 \pmod{55} \\ &= 46,656 \times 46,656 \times 46,656 \times 46,656 \times 46,656 \times 46,656 \times 46,656 \times 46,656 \times 46,656 \times 46,656 \pmod{55} \\ &= 16 \times 16 \times 16 \times 16 \times 16 \times 16 \times 16 \times 16 \times 16 \times 16 \pmod{55} \\ &= 16^2 \times 16^2 \times 16^2 \times 16^2 \times 16^2 \times 16 \pmod{55} \\ &= 36 \times 36 \times 36 \times 36 \times 36 \times 16 \pmod{55} \\ &= 967,458,816 \pmod{55} \\ &= 16 \pmod{55} \end{aligned}$$

The remaining encrypted data {10, 49, 21, 35} is calculated in the same way, and we record the results as follows:

$$\begin{aligned} 10^{33} \pmod{55} &= 10 \\ 49^{33} \pmod{55} &= 4 \\ 21^{33} \pmod{55} &= 21 \\ 35^{33} \pmod{55} &= 30 \end{aligned}$$

Thus, we would get the plaintext data represented in decimal notation as follows:

$$16 \quad 10 \quad 4 \quad 21 \quad 30$$

### STEP 2

Next, convert the decimal numbers you have obtained into 5-bit binary data:

|       |       |       |       |       |
|-------|-------|-------|-------|-------|
| 16    | 10    | 4     | 21    | 30    |
| ↓     | ↓     | ↓     | ↓     | ↓     |
| 10000 | 01010 | 00100 | 10101 | 11110 |

### STEP 3

To make the bits correspond with the character encoding in the table on page 169, divide the binary data into 6-bit units:

|           |           |           |           |               |
|-----------|-----------|-----------|-----------|---------------|
| 10000     | 01010     | 00100     | 10101     | 11110         |
| ↓ ↓ ↓ ↓ ↓ | ↓ ↓ ↓ ↓ ↓ | ↓ ↓ ↓ ↓ ↓ | ↓ ↓ ↓ ↓ ↓ | ↓             |
| 100000    | 101000    | 100101    | 011111    | 0 ← Eliminate |

Because the last 0 was used as padding, it won't fit into a 6-bit unit. You can eliminate it.

### STEP 4

Convert each 6-bit binary unit into an integer:

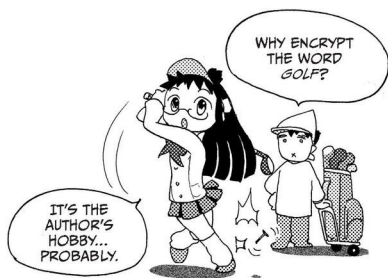
|        |        |        |        |
|--------|--------|--------|--------|
| 100000 | 101000 | 100101 | 011111 |
| ↓      | ↓      | ↓      | ↓      |
| 32     | 40     | 37     | 31     |

### STEP 5

Using the table on page 169, encode the integer data into characters:

|    |    |    |    |
|----|----|----|----|
| 32 | 40 | 37 | 31 |
| ↓  | ↓  | ↓  | ↓  |
| G  | O  | L  | F  |

Now you've completed the decryption process.



# PUBLIC-KEY ENCRYPTION AND DISCRETE LOGARITHM PROBLEMS



DO YOU GET RSA ENCRYPTION NOW?



HA... HA HA. OF COURSE...!

BY THE WAY, FOR PUBLIC-KEY ENCRYPTION,

YOU DON'T USE JUST INTEGER FACTORIZATION LIKE IN RSA ENCRYPTION, RIGHT? WHAT ELSE IS THERE TO IT?



YES, THAT'S RIGHT. I'LL ALSO GIVE A SIMPLE EXPLANATION OF ELGAMAL ENCRYPTION, WHICH HAS ITS FOUNDATION IN THE DISCRETE LOGARITHM PROBLEM.

UGH, MORE COMPLICATED VOCAB...

SIMPLE? YEAH RIGHT!

BUT IT REALLY IS!

FIRST READ THE FOLLOWING EXPLANATION...

AND LEARN ABOUT THE THEORETICAL FOUNDATIONS OF ENCRYPTION!

CREAK CREAK CREAK

## DISCRETE LOGARITHM PROBLEMS

Take another look at the modulo 7 power table. This table contains all the powers between 1 and 6 and shows the results of modding them by 7. We can express the possible results of a modulo 7 operation as a finite field comprising these members:

$$\{0, 1, 2, 3, 4, 5, 6\}$$

We'll represent this finite field in group notation. When we have a number  $n$  and a field of numbers related to  $n$ , we can represent the whole group as  $Z_n^*$ . In this case,  $n$  is 7, so we get  $Z_7^* = \{0, 1, 2, 3, 4, 5, 6\}$ . You'll notice that every number in this set is coprime to 7 except 0. We'll represent the group of coprimes in  $Z_n^*$  as  $Z_i$ .

You'll notice in the table to the right that in the row showing the power of 3 modulo 7, the values from 1 to 6 each appear in the row only once. This means  $Z_7^*$  with the exception of 0 is expressed in the power of 3 modulo 7. When we have a number  $n$ , if the power of  $a$  modulo  $n$  contains every coprime of  $n$  in  $Z_n^*$ , then  $a$  is called  $n$ 's *primitive root*.

Every prime number  $p$  has primitive roots, and the number of primitive roots is  $\phi(p - 1)$ . For modulo 7, the result of this equation would be

$$\phi(7 - 1) = \phi(6) = \phi(2 \times 3) = (2 - 1) \times (3 - 1) = 2$$

This means that modulo 7 has two primitive roots. Thus, aside from 3, there must be one other numerical value that is a primitive root, and, if we look at the  $a^b$  modulo 7 table, we can ascertain that it is 5. Where  $p$  is a prime number,  $a$  represents a primitive root modulo  $p$ , and  $k$  is a nonnegative integer but also fulfills the equation  $k \leq p - 1$ , we can express any member of  $Z_i$  as a modulo operation like so:

$$a^k = Z_i \pmod{p}$$

With some rearrangement of this equation, the exponent  $k$  of primitive root  $a$  can also be expressed as the following equation:

$$k = \log_a Z_i \pmod{p}$$

$a^b$  MODULO 7

| $a \backslash b$ | 1 | 2 | 3 | 4 | 5 | 6 |
|------------------|---|---|---|---|---|---|
| 1                | 1 | 1 | 1 | 1 | 1 | 1 |
| 2                | 2 | 4 | 1 | 2 | 4 | 1 |
| 3                | 3 | 2 | 6 | 4 | 5 | 1 |
| 4                | 4 | 2 | 1 | 4 | 2 | 1 |
| 5                | 5 | 4 | 6 | 2 | 3 | 1 |
| 6                | 6 | 1 | 6 | 1 | 6 | 1 |



When this is the case,  $k$  is known as a *discrete logarithm* with base  $\alpha$ .  
Let's quickly go over log notation. For example, the equation  $2^3 = 8$  is the same as

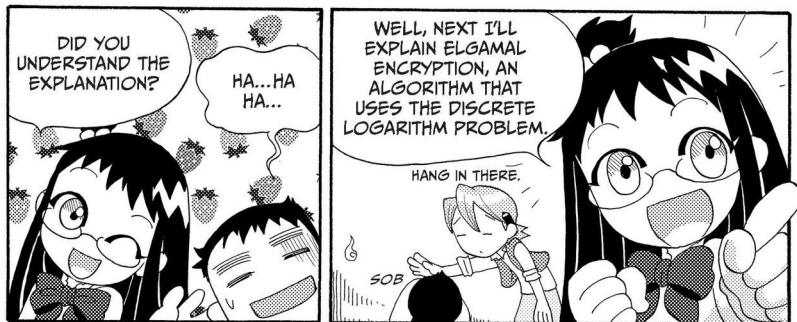
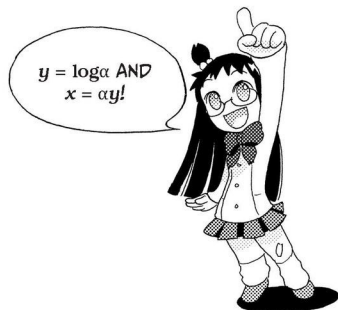
$$3 = 3 \log_2 2 = \log_2 2^3 = \log_2 8$$

This is similar to rephrasing the expression “2 to the power of 3 is 8” to be “to get 8, you must multiply 2 by itself 3 times.”

As is also explained on page 116, when you have the equation

$$\alpha^k = Z_i \pmod{p},$$

it isn't difficult to find  $Z_i$  when  $\alpha$ ,  $k$ , and  $p$  are known, but finding the discrete logarithm  $k$  is incredibly difficult even if you know  $\alpha$ ,  $Z_i$ , and  $p$ . This is the *discrete logarithm problem*.



## ENCRYPTION AND DECRYPTION IN ELGAMAL ENCRYPTION

In this cipher, the sender is Ruka, and the recipient is Ran.

1. Recipient Ran prepares a large prime number  $q$  and the primitive root  $\alpha$ .

WHO IS RAN?

THE GIRL FROM  
THE RAMEN SHOP!  
WE'VE BECOME  
FRIENDS!

2. Recipient Ran randomly chooses a private key  $d$ , performs the following calculation, and makes the public keys  $g$ ,  $\alpha$ , and  $q$  available to others.

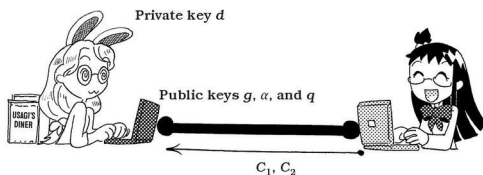
$$g = \alpha^d \pmod{q}$$



3. Sender Ruka selects a random number  $r$  and calculates  $C_1 = \alpha^r \pmod{q}$ .

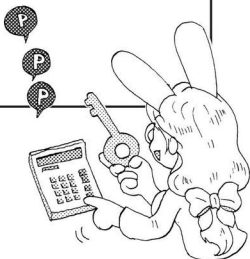
In addition, she calculates  $C_2 = m \times gr \pmod{q}$  for the plain-text message  $m$ .

4. Sender Ruka sends  $C_1$  and  $C_2$  to Ran.



5. The recipient Ran uses private key  $d$  to calculate the following equation and decrypt the message:

$$m = \frac{C_2}{C_1^d} \pmod{q}$$



WE HAVE  
 $C_1^d = (\alpha^r)^d = (\alpha^d)^r = \alpha^{rd} = g^r$   
 SO  
 $\frac{C_2}{C_1^d} = \frac{m \cdot g^r}{g^r} = m$   
 IS THE DECRYPTION  
 OF  $m$ .

WOW! WE'VE  
 DECRYPTED THE  
 PLAINTEXT  $m$ !

AMAZING!

SHARING A KEY IN ELGAMAL ENCRYPTION IS SIMILAR TO SHARING A KEY WITH THE DIFFIE-HELLMAN METHOD.

PINCH  
44

PAY  
ATTENTION!

3

1. Ruka and Ran share the large primary number  $p$  and the primitive root  $\alpha$ , neither of which is private.



2. Ruka selects a secret random number  $c$  and then sends  $\alpha^c \pmod{p}$  to Ran. On the other end, Ran selects a private random number  $d$  and then sends  $\alpha^d \pmod{p}$  to Ruka.

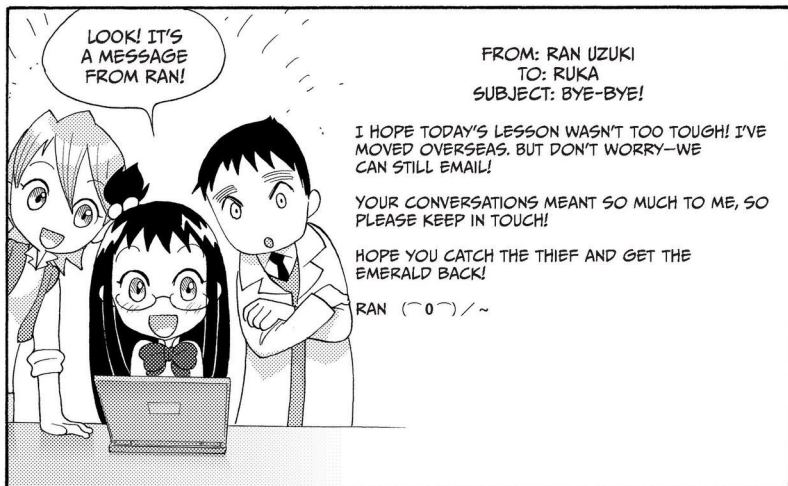
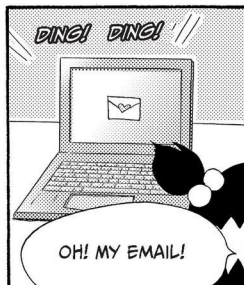
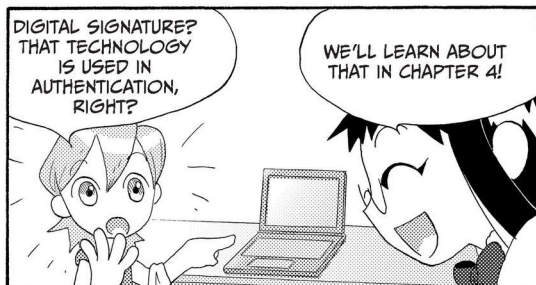


3. The shared key is  $\alpha^{cd}$ . Ruka obtains the shared key by calculating  $(\alpha^d)^c = \alpha^{cd} \pmod{p}$  from private key  $c$ , and Ran obtains the shared key by calculating  $(\alpha^c)^d = \alpha^{cd} \pmod{p}$  from private key  $d$ .

The two of them are able to share a key.

OH HH, THAT'S  
EASY TO  
UNDERSTAND!

ON TOP OF  
THAT, ELGAMAL  
ENCRYPTION IS  
REALLY STRONG!







## THE EXTENDED EUCLIDEAN ALGORITHM

The *Euclidean algorithm* derives the greatest common divisor of two natural numbers more efficiently than does base factorization. The procedure for identifying the greatest common divisor of two natural numbers  $a$  and  $b$  (where  $a > b$ ) using the Euclidean algorithm is as follows:

1. Divide  $a$  by  $b$  to find the remainder  $r$ .
2. When  $r = 0$ , the greatest common divisor is  $b$ , and the process is finished.
3. When  $r \neq 0$ , replace  $a$  and  $b$  with  $b$  and  $r$ , and perform the first step again.

In other words, repeat steps 1–3, and when you arrive at a remainder of 0, the divided number is the greatest common divisor.

For example, let's solve for the greatest common divisor of 1,365 and 77. Although the Euclidean algorithm deals with remainders, its proof is conventionally written without using modular arithmetic, so we'll do the same:

$$\begin{array}{ll} 1,365 = 17 \times 77 + 56 & 1,365 \div 77 = 17 \text{ with a remainder of } 56 \\ 77 = 1 \times 56 + 21 & 77 \div 56 = 1 \text{ with a remainder of } 21 \\ 56 = 2 \times 21 + 14 & 56 \div 21 = 2 \text{ with a remainder of } 14 \\ 21 = 1 \times 14 + \textcircled{7} & 21 \div 14 = 1 \text{ with a remainder of } 7 \\ 14 = 2 \times \textcircled{7} + 0 & 14 \div 7 = 2 \text{ with a remainder of } 0 \end{array}$$

So, the greatest common divisor is 7.

Next, we'll locate the greatest common divisor for the coprimes 20 and 17.

$$\begin{array}{l} 20 = 1 \times 17 + 3 \\ 17 = 5 \times 3 + 2 \\ 3 = 1 \times 2 + 1 \\ 2 = 2 \times 1 + 0 \end{array}$$

Of course, since the greatest common divisor is 1 for coprimes, the Euclidean algorithm seems unnecessary. However, the extended Euclidean algorithm is very useful to us.

The extended Euclidean algorithm is used to solve for the modular multiplicative inverse, which is one of the calculations required in RSA. It aims to solve for the coefficients of Bezout's identity, which is the following equation,

$$ax + by = d,$$

where  $d$  is the greatest common divisor of  $a$  and  $b$  ( $x$  and  $y$  are the coefficients we aim to solve for). To solve for  $x$  and  $y$ , we must perform a few more steps after Euclid's algorithm (hence the name extended Euclidean algorithm).

First, we'll transpose the first three equations into the following equations:

$$\begin{array}{l} 20 - 1 \times 17 = 3 \\ 17 - 5 \times 3 = 2 \\ 3 - 1 \times 2 = 1 \end{array}$$

Since 2 is equal to  $(17 - 5 \times 3)$ , as seen in the earlier equations, we'll take the last equation we created and substitute it for 2.

$$3 - 1 \times 2 = 1$$

$$3 - 1 \times (17 - 5 \times 3) = 1$$

Then manipulate the equation by distributing and combining like terms. The goal of these substitutions is to express everything on the left side in terms of  $ax + by$  where  $a = 20$  and  $b = 17$ .

$$3 - 1 \times (17 - 5 \times 3) = 1$$

$$3 + (-1 \times 17) + [-1(-5 \times 3)] = 1$$

$$(-1 \times 17) + 3 + (5 \times 3) = 1$$

$$(-1 \times 17) + (6 \times 3) = 1$$

Next, we'll substitute  $(20 - 1 \times 17)$  for 3:

$$(-1 \times 17) + 6 \times (20 - 1 \times 17) = 1$$

$$(-1 \times 17) + (6 \times 20) + (-6 \times 17) = 1$$

$$(6 \times 20) + (-7 \times 17) = 1$$

The result of this series of procedures is rewritten as follows:

$$20(6) + 17(-7) = 1$$

This is exactly what we were looking for. The value  $d = 1$  because  $a$  and  $b$  are coprimes,  $a = 20$  and  $b = 17$  as expected, and we now have the values of  $x$  and  $y$ .

The solution to the extended Euclidean algorithm here is  $x = 6$ ,  $y = -7$ .

## CALCULATING THE PRIVATE KEY IN RSA KEY GENERATION

Here we'll show how we got  $d = 33$  in our key generation example, which requires calculating inverse elements in a modulo operation.

In our example, we were trying to solve for the private key  $d$  and had this equation:

$$17d = 1 \pmod{40}$$

To solve this, you can rearrange the equation into Bezout's identity ( $ax + by = 1$ ) and use the extended Euclidean algorithm. We'll arrange the equation so that  $a = 17$ ,  $b = 40$ , and  $x = d$  to get

$$17x + 40y = 1$$

Now, to solve for  $x$ , let's first perform Euclid's algorithm:

$$40 = (2 \times 17) + 6$$

$$17 = (2 \times 6) + 5$$

$$6 = (1 \times 5) + 1$$

$$5 = (5 \times 1) + 0$$

Now we'll perform the extended Euclidean algorithm. Let's start with the second-to-last equation from our calculations for Euclid's algorithm:

$$6 - 1 \times 5 = 1$$

Now substitute the third-to-last equation into the second-to-last equation:

$$6 - 1 \times (17 - 2 \times 6) = 1$$

Now substitute in the first equation as well:

$$(40 - 2 \times 17) - 1 \times (17 - 2 \times (40 - 2 \times 17)) = 1$$

Group the numbers into similar terms. In this case, group them into 17s and 40s:

$$-7 \times 17 + 3 \times 40 = 1$$

Given that we had  $17x + 40y = 1$  and were solving for  $x$ , we can see that our private key  $d = x = -7 = 33 \pmod{40}$  became  $-7 + 40 = 33$ .

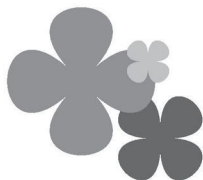
Now that you've completed this hands-on example, you should have the confidence to solve for other private keys for RSA key generation!

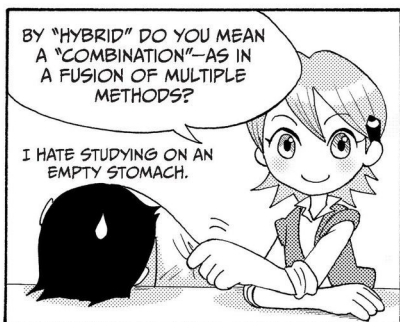
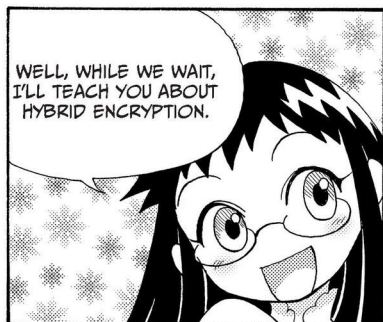
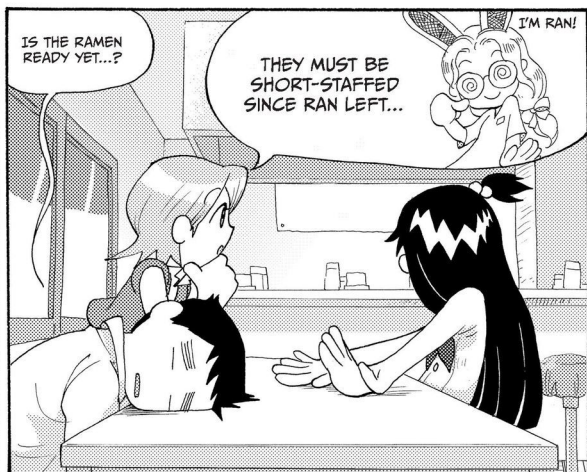
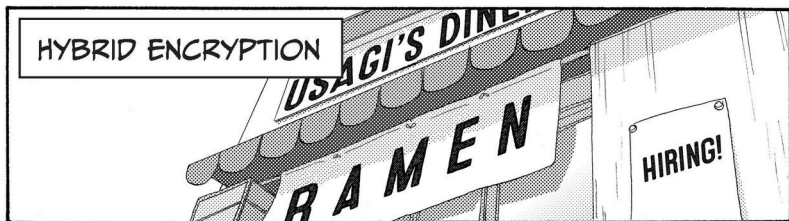


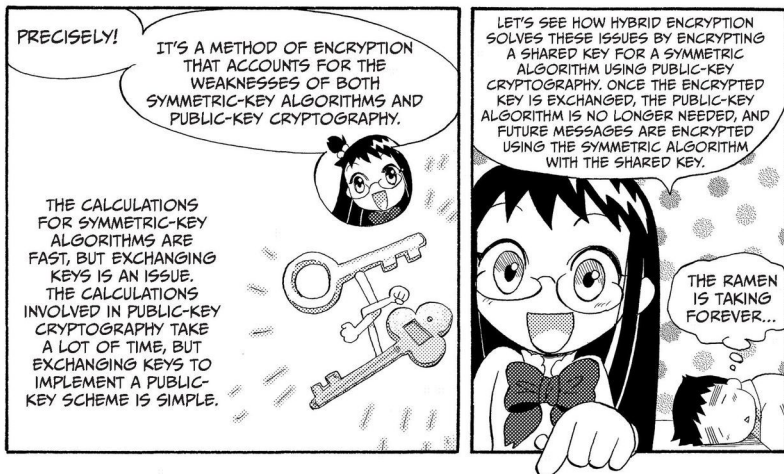


# 4

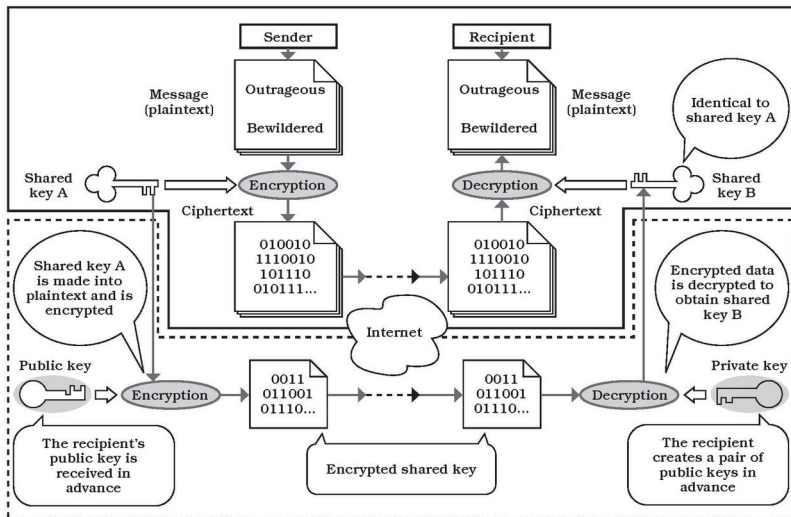
## PRACTICAL APPLICATIONS OF ENCRYPTION







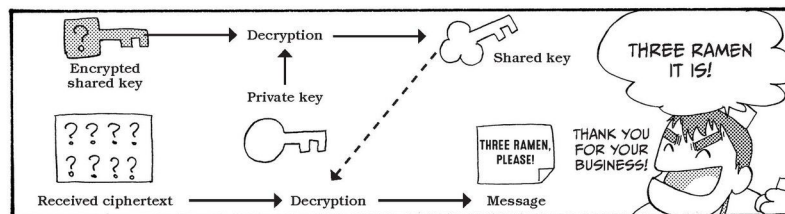
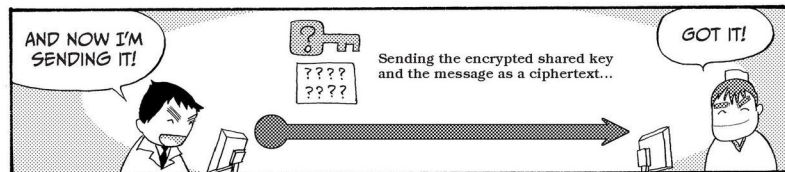
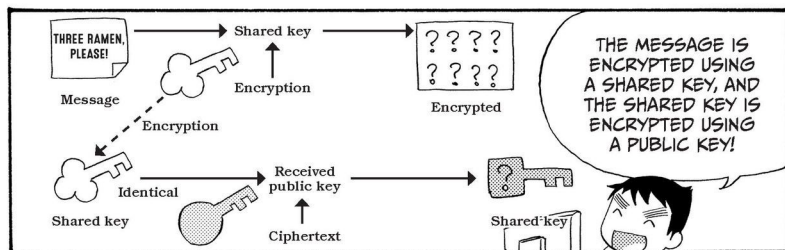
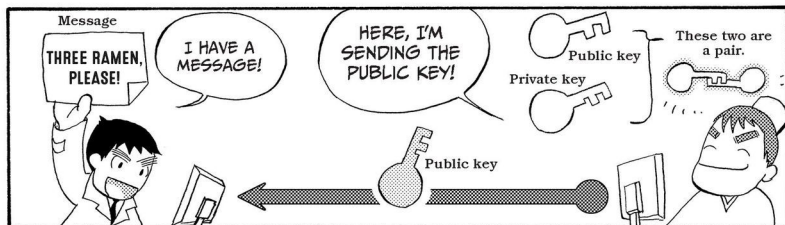
Symmetric algorithm

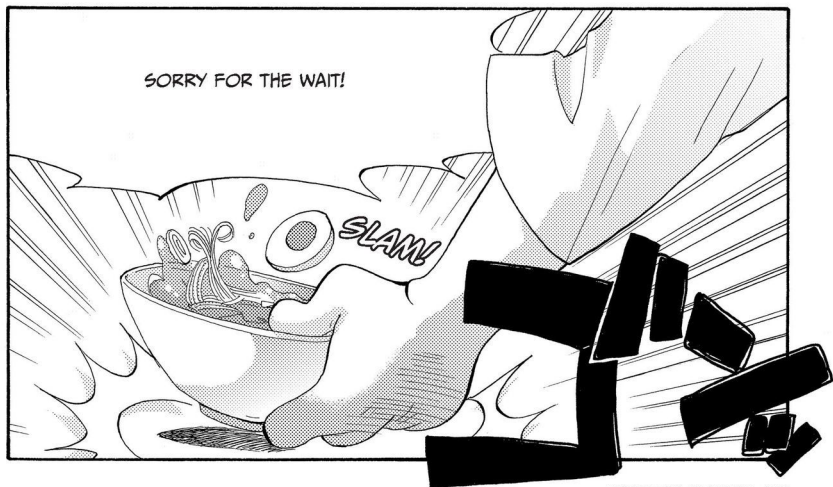
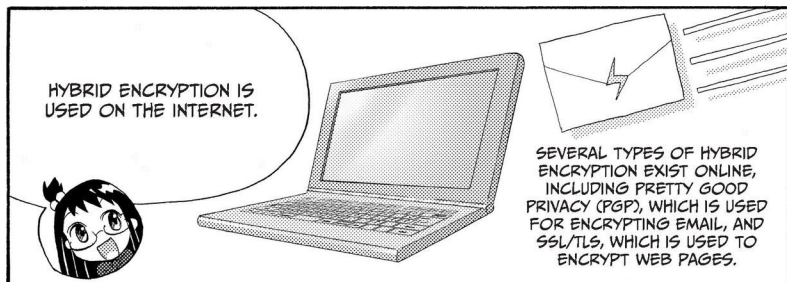


Public-key encryption process

As demonstrated in the previous figure, public-key encryption is used only to encrypt and decrypt the shared key so that the symmetric-key algorithm can be used to encrypt and decrypt the actual messages that are exchanged. Since the shared key is encrypted using a public key, the greatest weakness of a shared key—the delivery of the key—is avoided.

So, now let's look at a real-life example of encryption using ramen orders.





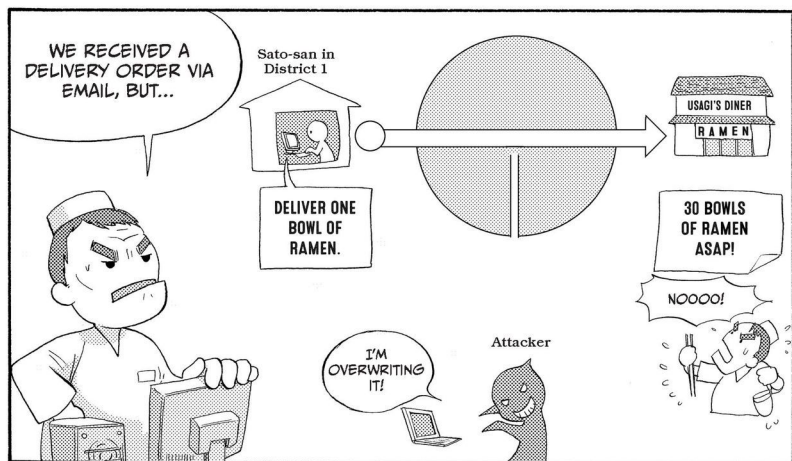


## HASH FUNCTIONS AND MESSAGE AUTHENTICATION CODES



### FALSIFICATION





## TAMPERING COUNTERMEASURES

TO STOP TAMPERING,  
YOU SHOULD USE HASH  
FUNCTIONS!

hash

HASH? IS THAT LIKE  
BEEF HASH?

HOLD THE  
DROOL. I DON'T  
THINK THERE'S  
BEEF INVOLVED.

"HASH" REFERS TO  
SOMETHING THAT'S  
CHOPPED UP.

INSTEAD OF BEEF, HASH  
FUNCTIONS CHOP AND MANGLE  
MESSAGES INTO TINY PIECES TO  
PRODUCE HASH VALUES.

SOUNDS  
DELICIOUS.

OOH, I SEE!

WHAT'S A HASH  
VALUE?

MORE IMPORTANTLY,  
IS IT EDIBLE?

IT'S A VALUE CALCULATED FROM THE  
MESSAGE. IT'S LIKE A FINGERPRINT  
YOU WOULD USE TO IDENTIFY A  
PERSON IN A CRIMINAL INVESTIGATION.

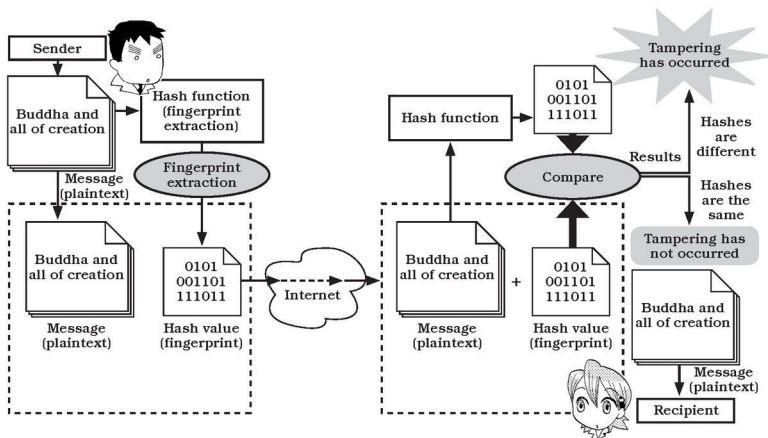
THEY'RE  
NOT FOOD!

WE USE HASHES  
TO CHECK THAT  
MESSAGES HAVEN'T  
BEEN FALSIFIED.

## HASH FUNCTIONS

A hash value identifies a message, like a fingerprint. The hash value is a small, fixed size, whereas the input is often larger than the hash. A hash function is used to calculate a hash value from a message.

The sender transmits a hash value with a message to guarantee the message's integrity. A message has *integrity* when the recipient is assured that the message hasn't been falsified. The recipient recalculates the hash using the same hash function as the sender and then compares that calculated hash to the hash attached to the message. If the values are the same, the message hasn't been falsified. This process, shown in the following figure, is called performing a *checksum*.

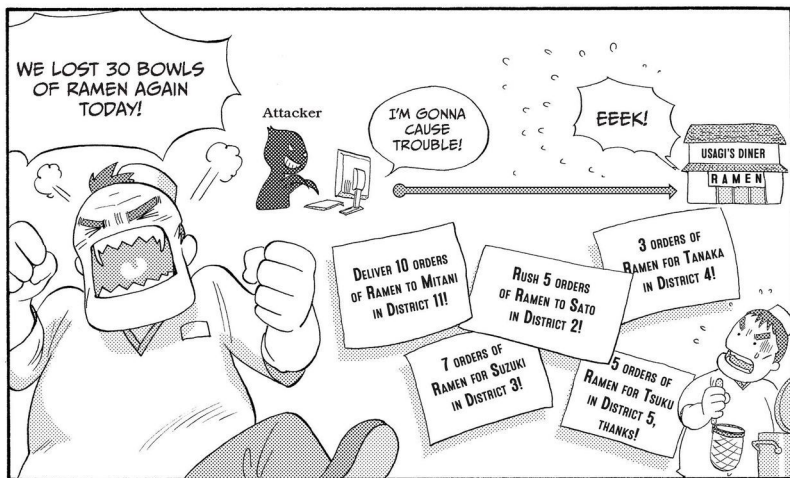


A hash function is a one-way function. The original message can't be derived from the hash value. This property is referred to as *irreversibility*.

Hash functions need to be resistant to *collision attacks*, which guess the hashes for messages. Because a hash needs to be unique to a message, the hash function can't produce the same hash for different messages. When a hash function fulfills this requirement, it has *strong collision resistance*. In addition to being resistant to hash collisions, a hash function shouldn't produce similar values for similar messages. For example, if we were to give the input 01001 to a hash function, it should output a different value than if we used 01000 as the input, which differs by only 1 bit. Real-life hash functions have a larger input than output, which means collisions are unavoidable. Therefore, they don't have strong collision resistance. However, hash functions can be developed to make finding an input that matches an output as difficult as possible. This is accomplished by using a large output and sets of calculations that are thought to be resistant to reverse engineering. Some hash functions developed with these requirements in mind are MD5, SHA-1, SHA-256, and SHA-512, but these functions are not foolproof. For example, researchers discovered that MD5 and SHA-1 are vulnerable to collision, so they are no longer safe to use.



## IDENTITY FRAUD





## IDENTITY FRAUD COUNTERMEASURES

YOU CAN USE  
A MESSAGE  
AUTHENTICATION  
CODE!

MAC: MESSAGE AUTHENTICATION CODE

MESSAGE  
AUTHENTICATION?

WHAT'S THAT?

WHEN YOU HAVE A  
MESSAGE FROM THE  
ACTUAL SENDER...

DISTRICT 4:  
2 RAMEN!

SATO IN  
DISTRICT 1:  
1 RAMEN!

AIDA IN DISTRICT 6:  
RAMEN AND  
GYOZA

NAKAI IN  
DISTRICT 3:  
3 RAMEN!

TANAKA IN  
DISTRICT 2

IT AUTHENTICATES THAT  
THE RECEIVED MESSAGE  
HAS COME FROM THE  
CORRECT PERSON

AND ISN'T FROM A  
FRAUDULENT SENDER  
ENGAGED IN IDENTITY  
THEFT.

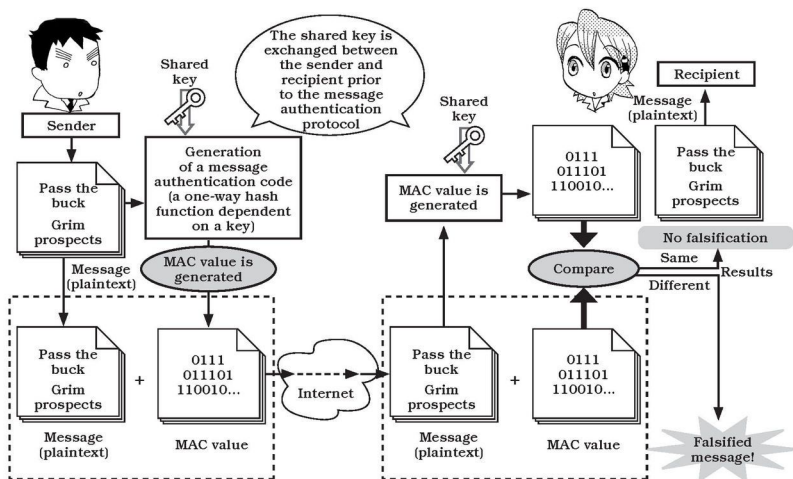
WHOA!

HOW DOES IT  
AUTHENTICATE  
MESSAGES?

## STRUCTURE OF MESSAGE AUTHENTICATION CODES

A message authentication code is a procedure that confirms a message's authenticity and certifies a message. Let's look at the following figure to learn how a message authentication code can be produced.

The sender transmits a message and the MAC value (also called a tag) derived from their message. The MAC value is used to perform a checksum in the same way as with a hash value.



The sender delivers a message with its corresponding MAC value to a recipient. The recipient calculates a MAC value from the message they received and compares their MAC with the original MAC value from the sender. If the two MAC values match, the recipient is assured that the integrity of the message is intact and that the message is from the sender.

**NOTE:** The recipient knows that the message is from the sender because it is assumed that only the sender and recipient have the key. If the recipient didn't create the tag (the MAC value), then it must be from the sender.

Think of a message authentication code as a one-way hash function that is associated with the shared key. MAC values are similar to hash functions because they possess the same mechanisms and confirm the authenticity of a message by calculating and comparing values on the sender's and recipient's ends.

When the two parties calculate the MAC value, they use a key that is shared only between themselves. By calculating the MAC value from the message, the receiving party can be assured that the sender of the message is in fact the person who has the same shared key.

MAC values provide integrity for a message. When the two MAC values are identical, the message has not been tampered with en route from the sender. When the two MAC values differ, the message has potentially been falsified, and the sender may not be the one who shared the key.

A message authentication code works in this way, but as with public-key encryption, there are issues with the security of the shared key.

Message authentication codes are also used for SSL/TLS, which is used by international banks for money transfers and in online shopping.



## REPUDIATION

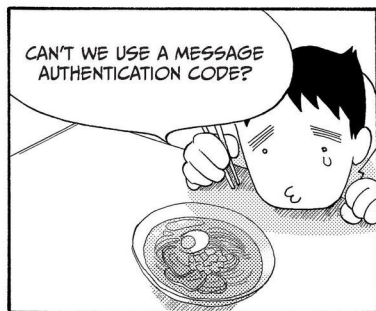
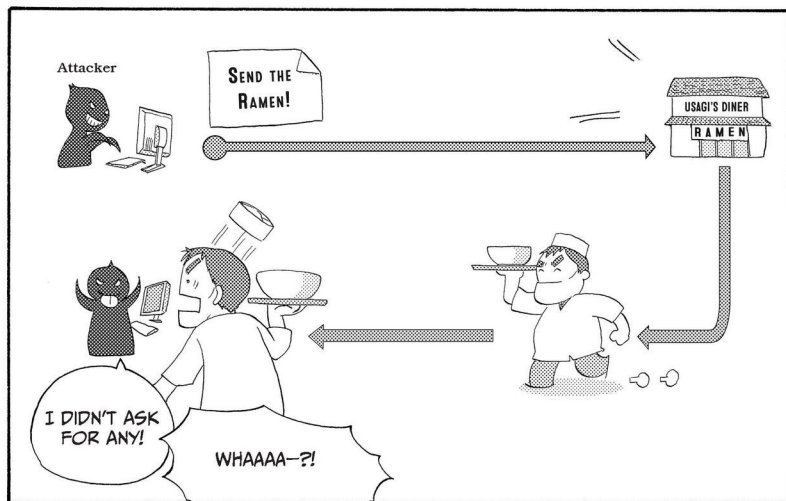
MISTER, NOW YOU  
CAN REST EASY!

NOW LET'S  
HAVE SOME  
RAMEN!

NO—WAIT!

IS THERE STILL  
AN ISSUE?

WHAT WOULD  
YOU DO IN THIS  
SITUATION?



## DRAWBACKS OF MESSAGE AUTHENTICATION CODES

*Repudiation* is the ability to deny being the sender of a message. For example, a message and MAC value are sent from A to B, and afterward A claims, "I didn't send this message to B. B made this up." There's no way to disprove A's statement, and even if B enlisted the help of a third party to find out the truth, this third party wouldn't have a way to determine whether the message and MAC value were generated by A or by B.

When a message is sent from A to B, B can't verify to a third party C that the message was sent from A. This is because the message and MAC value can be generated by either A or B. In other words, C is unable to determine whether the MAC value was generated by A or B.





## DIGITAL SIGNATURES



## REPUDIATION MEASURES

HOW DO YOU  
DEFEND AGAINST  
REPUDIATION?

YOU USE A DIGITAL  
SIGNATURE!

A DIGITAL SIGNATURE IS THE  
OPPOSITE OF ENCRYPTION IN  
PUBLIC-KEY CRYPTOGRAPHY.

DOING THIS WILL ALSO  
ALLOW THIRD PARTIES TO  
VERIFY YOUR IDENTITY.



HOW DOES  
IT WORK?

MESSAGE  
AUTHENTICATION CODES  
PROVIDE INTEGRITY IN  
PRIVATE-KEY/SYMMETRIC  
ENCRYPTION, WHEREAS  
A DIGITAL SIGNATURE  
PROVIDES INTEGRITY  
IN PUBLIC-KEY  
ENCRYPTION.

THE TABLE SHOWS  
ENCRYPTION AND DIGITAL  
SIGNATURES IN PUBLIC-  
KEY CRYPTOGRAPHY.



LET'S CHECK  
OUT THE  
STRUCTURE  
OF A DIGITAL  
SIGNATURE!



|                           |                                      |                   |                                          |
|---------------------------|--------------------------------------|-------------------|------------------------------------------|
| <b>Encrypting message</b> | Encryption using sender's public key | <b>Ciphertext</b> | Decryption using recipient's private key |
| <b>Digital signature</b>  | Decryption using sender's public key | <b>Signature</b>  | Encryption using sender's private key    |

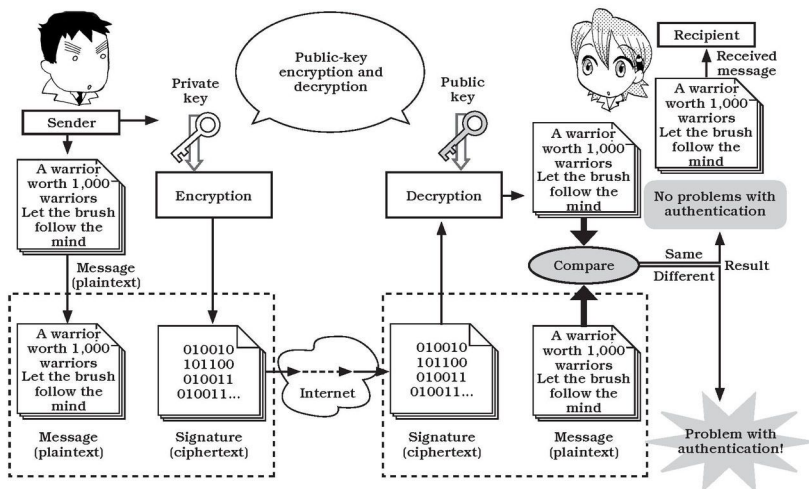
## THE WORKINGS OF A DIGITAL SIGNATURE

When using a digital signature, the sender signs a message that has been encrypted using their own private key. This signature is sent to the recipient along with the message.

The recipient decrypts the signature using the sender's public key to obtain the message. Then the recipient compares the decrypted message to the sent message.

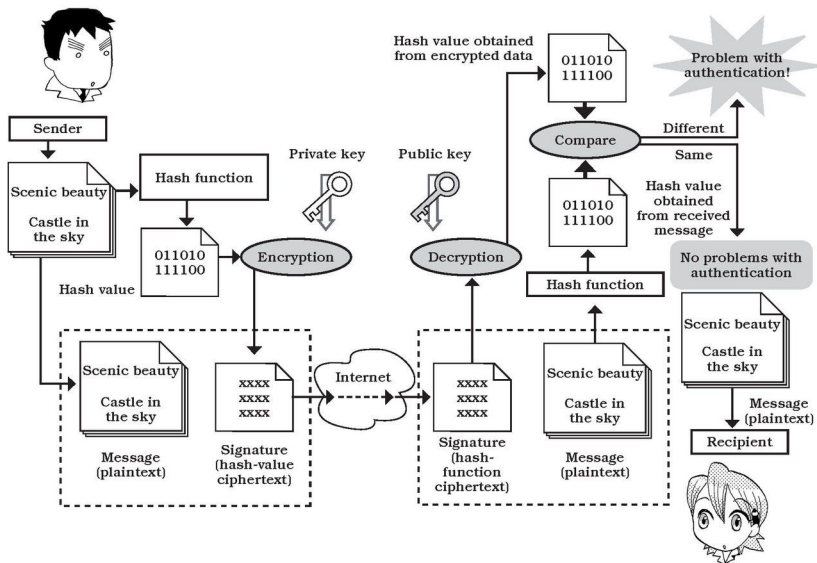
If the two are the same, the message is verified as authentic, and the sender is thought to be legitimate. Because the sender's public key is used for decryption, a third party can also verify the signature. When a third party is able to verify that a message came from a specific sender, we have nonrepudiation—the sender can't deny they sent a message.

Let's look at some examples of how a digital signature works. The following figure has been simplified to make the general concept of a digital signature easier to understand.



In this figure, the message shown is immediately encrypted and signed. In reality, it takes time to sign an entire message with public-key cryptography.

It is common to first use a hash function to turn the message into a hash value and then convert the hash into a signature, as shown in the following figure.



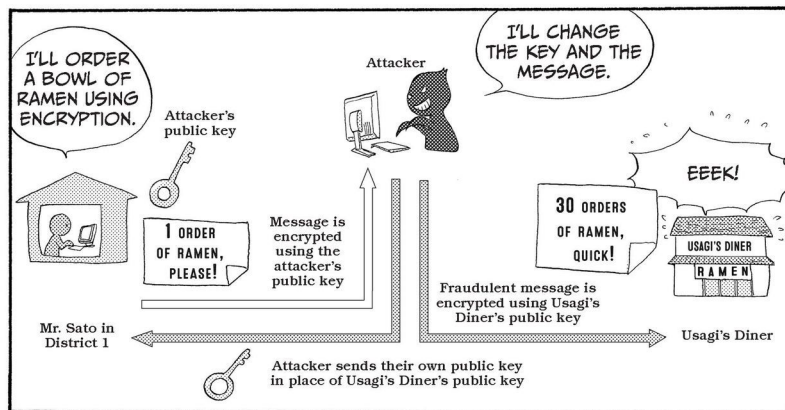
A digital signature is also used to generate a server certificate that authenticates the legitimacy of an SSL/TLS server. A certificate is something that appends a digital signature to a public key (in this case, the server's public key). Certificates also append digital signatures to software downloads to protect software from being falsified.

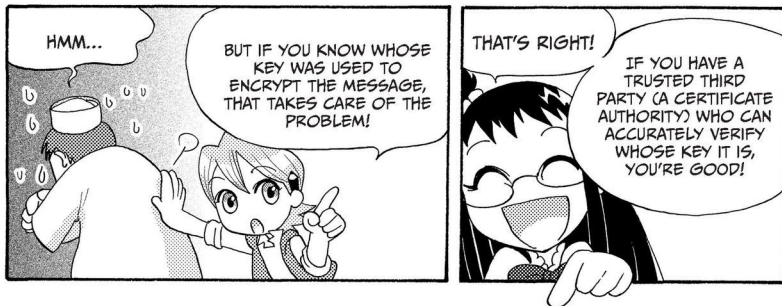


### MAN-IN-THE-MIDDLE ATTACK

Let's say we have a sender A and a receiver B. We'll have Mr. Sato of District 1 be A and Usagi's Diner be B. Sender A corresponds with B using encryption, but first he has to receive B's public key. A person in the middle, an attacker, intercepts the public key as it is transmitted from B to A and then sends their own public key to A instead.

Since the encrypted message sent by A has been encrypted using the attacker's public key, the attacker can decode the message using their own private key. The attacker alters the contents of the message, encrypts it using B's public key, and sends it. Once the attacker is done, B has no idea that the message was tampered with and has no way to check for falsification. Let's see what this looks like.





## USING CERTIFICATES TO PROTECT AGAINST MAN-IN-THE-MIDDLE ATTACKS

A *certificate* attaches the digital signature of a public key and that key to an identity. A certificate is published by a *certificate authority*. When someone wants to make a public key available, they register their public key using a certificate authority and simultaneously request publication of the certificate.

Based on this request, the certificate authority verifies the authenticity of the user, and, if the public key meets the certificate authority's standards, the certificate authority generates a certificate for the public-key and digital-signature set. Sometimes the user creates the public- and private-key set, while other times the certificate authority generates these during registration.

The process of verifying the certificate guarantees that the public key indeed belongs to a specific user. The certificate authority, a trusted third party, verifies that the public key is correct on behalf of the user. Now let's look at the procedure shown in the next figure and outlined in these six steps.

1. User A requests that the certificate authority publish the user's public key.
2. After the certificate authority confirms the identity of User A, the certificate authority produces a digital signature for User A's public key, and the certificate is published. The published certificate affixes a digital signature to User A's public key by means of the certificate authority. The certificate is usually made up of the public key of User A (PKA) and the digital signature of PKA.
3. The certificate authority stores the certificate in a *repository* (a data storehouse).
4. User B downloads User A's certificate from the repository.
5. User B decrypts the digital signature using the certificate authority's public key.
6. The decrypted key is compared with the public key attached to the certificate. If the two keys are the same, the public key included with the certificate is verified as belonging to User A.

By means of the certificate-issuing procedure, User B can obtain User A's verified public key. By using User A's secured public key, User B can verify that

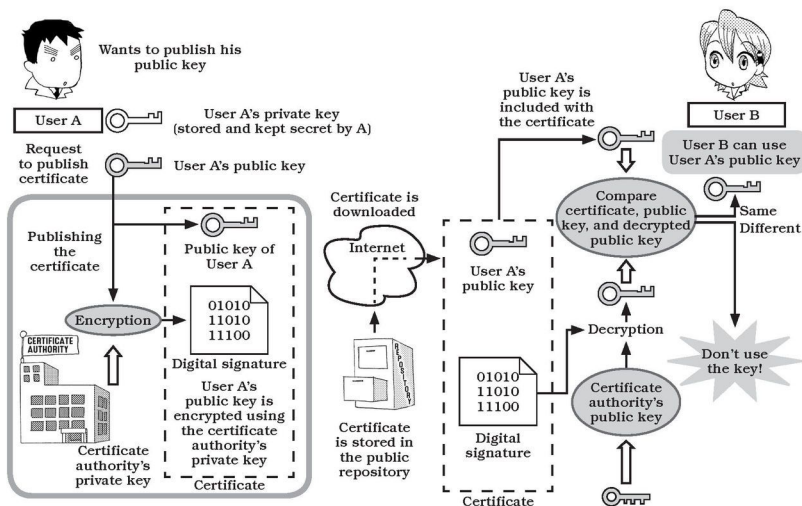


the message was encrypted with User A's private key and affixed with a digital signature.

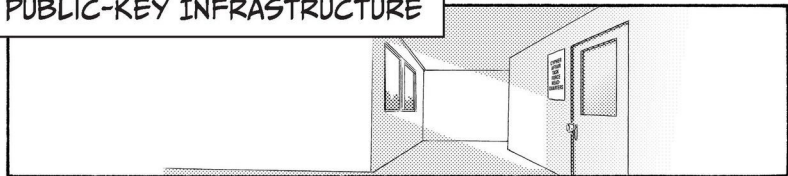
The validated message will simultaneously meet the following three conditions:

1. The message has not been falsified.
2. The message has not been generated by a third party posing as User A.
3. User A can't repudiate the message. In other words, User A can't deny they generated the message.

As long as the legitimacy of a public key is verified, any message affixed with the public key's digital signature is assured to meet these three conditions. Let's finish up by discussing the mechanisms behind public-key infrastructure (PKI).



## PUBLIC-KEY INFRASTRUCTURE



WE'RE ALMOST  
DONE STUDYING  
ENCRYPTION!



BUT CAN WE  
REALLY RELY  
ON THOSE  
CERTIFICATES?



HOW CAN WE KNOW A  
CERTIFICATE ACTUALLY  
WAS PUBLISHED BY A  
CERTIFICATE AUTHORITY?

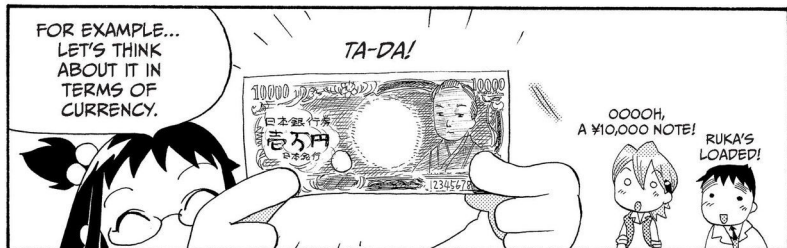
CAN WE REALLY TRUST  
THE CERTIFICATE  
AUTHORITY?

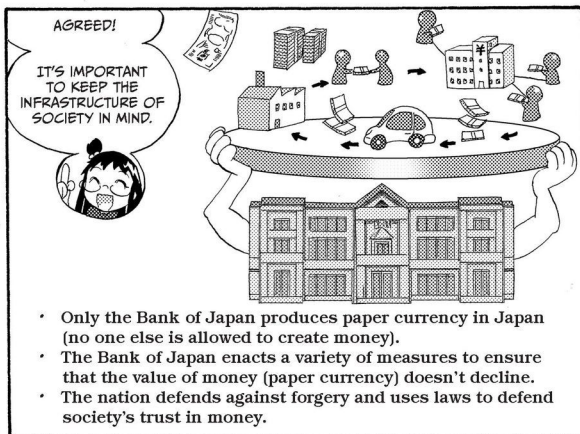
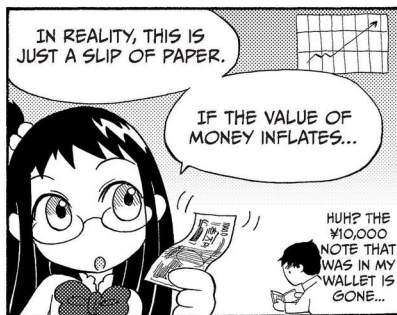
I STILL  
HAVE SOME  
DOUBTS.

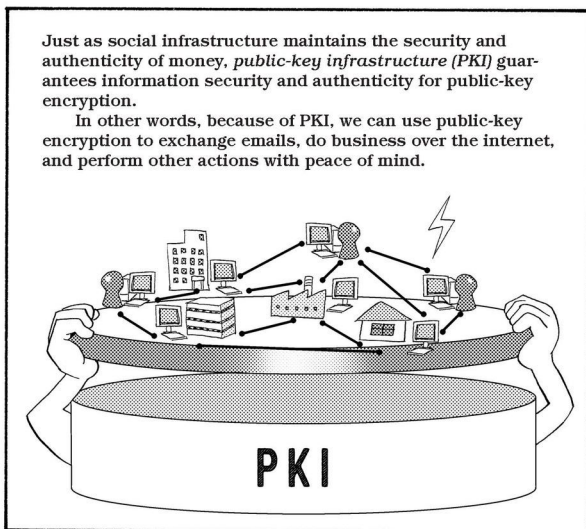
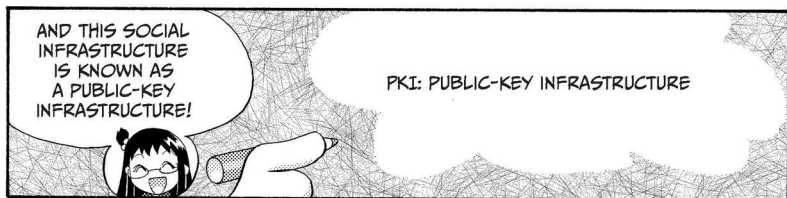
HOW DO WE KNOW  
INFORMATION IS  
AUTHENTIC?



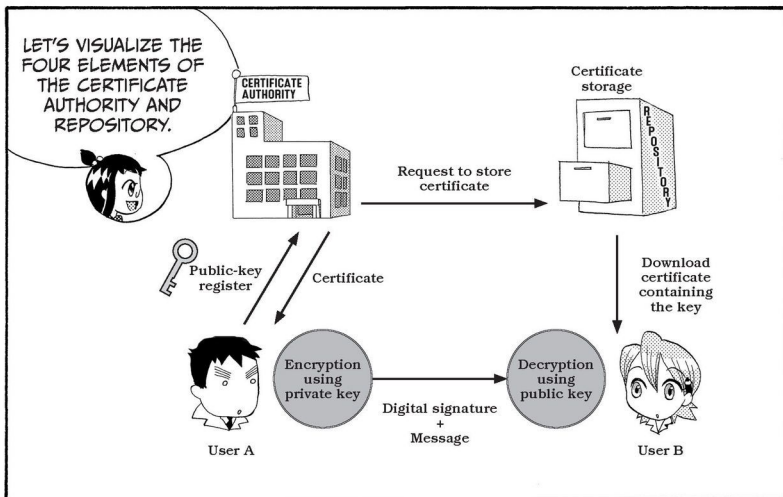
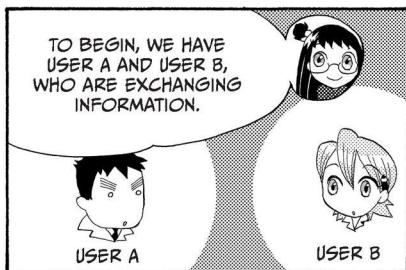
FOR EXAMPLE...  
LET'S THINK  
ABOUT IT IN  
TERMS OF  
CURRENCY.





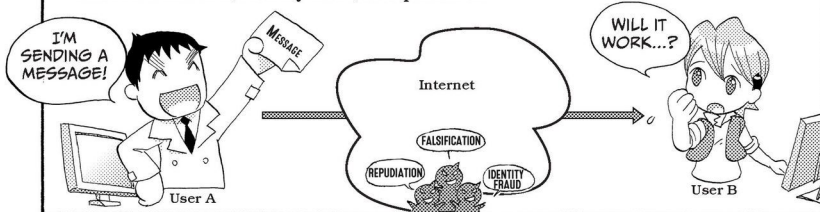




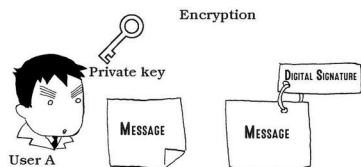




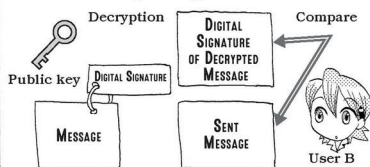
User B wants to receive User A's message without fear of falsification, identity fraud, or repudiation.



Using his own private key, User A attaches a digital signature he has generated to the message. Then he sends the message to User B.



Using User A's public key, User B verifies the digital signature on the received message. If the decryption of the digital signature with User A's public key matches the message sent, the message is legitimate.



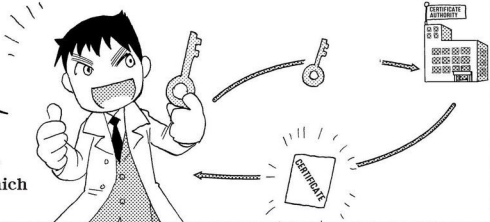
BUT HOW DO WE KNOW THIS PUBLIC KEY ACTUALLY BELONGS TO USER A?

IT'S SUSPICIOUS!

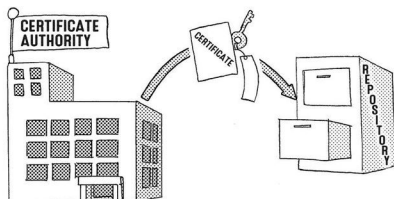
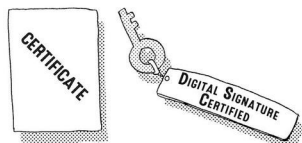
HMM, HOW WOULD YOU VERIFY THAT?

THAT'S IT! THE TRUSTED CERTIFICATE AUTHORITY CAN VERIFY THE PUBLIC KEY!

User A registers their public key with the certificate authority, which publishes a certificate.

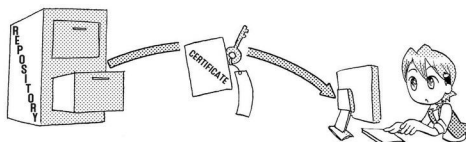


The certificate is composed of User A's public key and the certificate authority's digital signature.

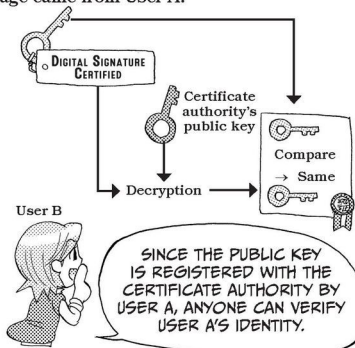


The certificate authority stores the certificate in a repository.

User B downloads User A's certificate from the repository.

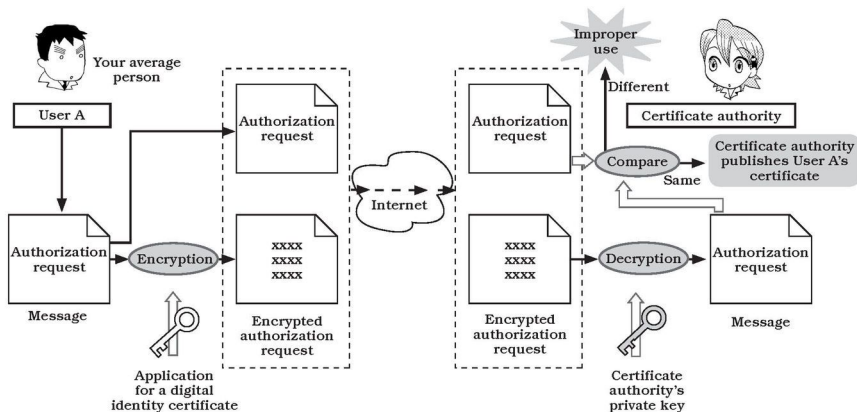


User B decrypts the digital signature included in User A's certificate. User B then compares the public key and digital-signature key. If the two keys are the same, User B has verified that the message came from User A.

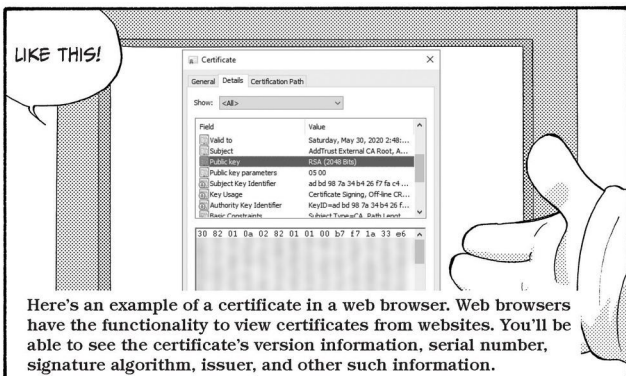
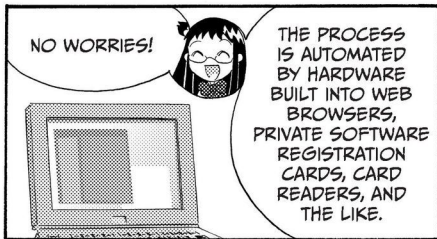


When verification is complete, we know that User A's public key is legitimate and the message User B received is also legitimate (in other words, we've ruled out falsification, identity fraud, and repudiation).





When the decryption of the received application and the public key are verified, the certificate authority will grant User A the certificate.





WHA-WHAAAAAAAT?!



FROM: RAN (MS. CYPHER)  
TO: RUKA  
SUBJECT: ARE YOU WELL?

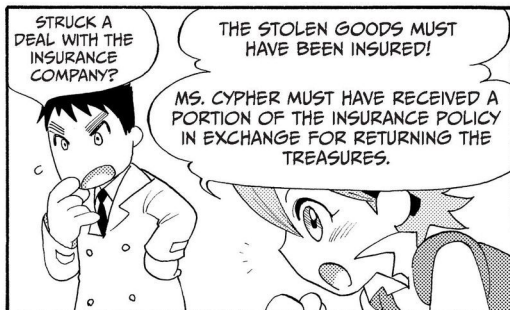
I'M RETURNING THE SMILING MADONNA  
AND THE PRECIOUS EMERALD.

(I'VE STRUCK A DEAL WITH THE  
INSURANCE COMPANY.)

BY THE WAY, I'M CURRENTLY IN THE  
COUNTRY OF LIBERTY, AND I HAVE A BIG  
JOB COMING UP VERY SOON!

HERE'S A HINT, IN BINARY:  
00000010 00001100 00010101  
00011100 00011101 00000110  
00010010

WELL THEN, SEE YA!

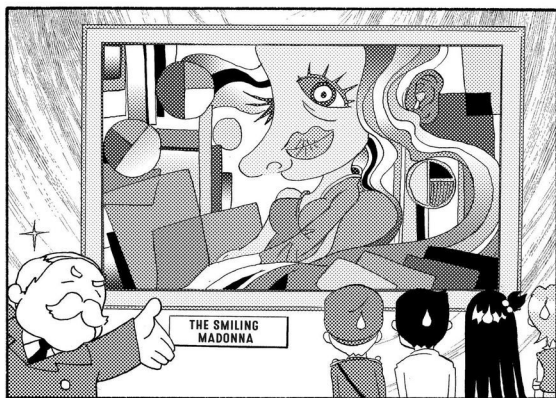






SEVERAL DAYS LATER...





FROM HERE ON, LET'S BE ATTENTIVE WHEN IT  
COMES TO INFORMATION SECURITY SO WE CAN  
BUILD A SAFER SOCIETY!



# ZERO-KNOWLEDGE INTERACTIVE PROOF

In order to use a credit card, you need to share information to verify the authenticity of the card, but credit card theft is common. If your credit information is stolen while you are authenticating your card, you could be charged for items you don't recall purchasing (because you didn't purchase them!). In the same way, you risk revealing private information to outsiders whenever you need to verify your identity. Because of these risks, we need methods that ensure confidentiality can't be breached (also known as a *zero-knowledge protocols*) and methods that verify a person's identity to others (also known as *authenticity*).

In an effort to respond to these necessities, Goldwasser, Micali, and Rackoff introduced the concept of the *zero-knowledge interactive proof* in 1985. The zero-knowledge proof is a means by which a third party (for example, a credit card company) can verify the authenticity of an individual's card while ensuring that the private information attached to a card is not leaked. For example, a card has a randomized decimal password of more than 100 digits. To authenticate a card, a company or individual charging to the card would need to verify that the password is authentic without the ability to see the card's password. This might seem impossible, but we can do this using the mathematical theory behind encryption. Let's look at how this method works in detail by breaking the process down into two stages: the preparatory stage and the implementation stage.

## PREPARATORY STAGE

The setup of a zero-knowledge proof first requires an honest *verifier* (such as a credit card company). Let's look at an example.

*CREATE A COMPOSITE NUMBER  $N$  THAT IS PUBLICLY AVAILABLE TO ALL USERS*

The verifier prepares two prime numbers ( $p$ ,  $q$ ) and calculates their product, the composite number  $N$ . That is, they use the following equation:

$$N = pq$$

The prime numbers  $p$  and  $q$  are secret. In actual practice, a proof would use large prime numbers comprising around 80 digits, but here, we'll use the two-digit prime numbers  $p = 13$  and  $q = 19$  for simplicity's sake. The product of these two prime numbers is

$$N = 13 \times 19 = 247$$

This is a three-digit composite number, but as noted, the factorization of  $N$  in practice would generate a number so large that it would be impossible for any computer to calculate. This very large composite number  $N$  is made available to all users.

*REGISTER EACH USER'S ID WITH THE VERIFIER*

Each user has an ID that is a publicly available numerical value associated with each user's public key. These IDs enable users to engage in one-to-one correspondence. Each user registers their ID with the verifier. In this book, we'll identify IDs with subscripts. For example, User A's ID is  $ID_A$ .

The verifier calculates the user's registered ID modulo the square root of  $N$ . When dealing with real numbers, calculating a square root is simple; however, because we need to use integers, the square root calculation is complex unless the prime numbers  $p$  and  $q$  of composite number  $N$  are known. The zero-knowledge proof is built upon this computationally complex calculation.

In this example, only the verifier knows the prime numbers 13 and 19, and since only the verifier can calculate the square root of each ID registered by the users, confidentiality is not breached. Let's look at an example in which  $ID_A$  is 101. In this case, the square root is 71.

$$\sqrt{101} \pmod{247} = 71$$

When we calculate in reverse and square 71, we get  $71^2 \pmod{247} = 101$ . The value 71 is User A's private key ( $S_A$ ), which is secretly delivered to User A. Generally, the following relationship exists between  $ID_A$  and private key  $S_A$ :

$$\begin{aligned}\sqrt{ID_A} \pmod{N} &= S_A \\ (S_A)^2 \pmod{N} &= ID_A\end{aligned}$$

In practice, the private key will be a number of more than 100 digits, so the private key isn't a number that User A can memorize. However, the purpose of the private key ( $S_A$ ) is not to verify the identity of User A but rather to verify the authenticity of User A's card. Therefore, User A doesn't need to memorize  $S_A$  as one would memorize the PIN for an ATM card. Now that User A has a number to verify that their card is authentic, let's look at how User A would use this card.

## IMPLEMENTATION STAGE (VERIFICATION PROCESS)

Now we'll demonstrate the verification process for when a *prover* such as User A wants to authenticate their identity to User B. If User A is trying to buy something with their credit card, this process shows that the card User A possesses is genuine.

### USER A SENDS A VERIFICATION REQUEST TO USER B

First, User A selects a random number  $r_A$ , squares it, and divides it by the composite number  $N$  to find the remainder,  $y_A$ . In equation form, the remainder  $y_A$  is calculated as follows:

$$y_A = (r_A)^2 \pmod{N}$$

The remainder  $y_A$  is sent to User B. Say for example that User A chooses the random number 50. The remainder  $y_A$  is calculated as follows:

$$y_A = 50^2 = 2,500 = 30 \pmod{247}$$

Thus, User A sends 30 to User B.

Next, User A takes the product of the private key  $S_A$  they received from the verifier and the random number  $r_A$ . Then User A divides the product by the composite number  $N$  to find the remainder,  $z_A$ .

$$z_A = S_A r_A \pmod{N}$$

Once User A calculates  $z_A$ , they send the value to User B. If we use the example where the random number  $r_A$  is 50, we get the following equation:

$$z_A = 71 \times 50 = 92 \pmod{247}$$

And 92 is sent to User B.

*USER B VERIFIES USER A'S AUTHENTICITY*

User B squares the  $z_A$  they received from User A and divides by the composite number  $N$  to find the remainder  $v_A$ .

$$v_A = (z_A)^2 \pmod{N} = (S_A r_A)^2 \pmod{N}$$

In our example,  $z_A = 92$ , so we get the following equation:

$$v_A = 92^2 = 8,464 = 66 \pmod{247}$$

Next, User B calculates the value of  $v_A$  divided by  $y_A$ , which User B received from User A earlier in the process.

$$w_A = \frac{v_A}{y_A} \pmod{N} = v_A \times (y_A^{-1}) \pmod{N}$$

All of the calculations are operations using the composite number  $N$ , and  $y_A^{-1}$  expresses the inverse element (reciprocal) of  $y_A$ . This means that  $y_A^{-1}$  is a value that satisfies the following equation:

$$y_A \times (y_A^{-1}) = 1 \pmod{N}$$

In this example,  $v_A = 66$ ,  $y_A = 30$ , and  $y_A^{-1} = 30^{-1} \pmod{247} = 140$ ; therefore, we can calculate  $w_A$  as follows:

$$w_A = \frac{66}{30} \pmod{247} = 66 \times 30^{-1} \pmod{247} = 66 \times 140 \pmod{247} = 101$$

Lo and behold, this is User A's ID ( $ID_A$ ).

Using this process, User B can verify the authenticity of User A's identity. We can find  $ID_A$  using this process because squaring User A's private key  $S_A$  yields  $ID_A$ . That is to say, we can derive  $(S_A)^2$  using the previous equations as follows.



$w_A$  is found by the equation  $w_A = v_A / y_A \pmod N$ .  
Recall the equations found for  $v_A$  and  $y_A$ :

$$v_A = (z_A)^2 \pmod N = (S_A r_A)^2 \pmod N$$

$$y_A = (r_A)^2 \pmod N$$

Substitute these into the equation for  $w_A$  to get

$$w_A = \frac{(S_A r_A)^2}{(r_A)^2} = (S_A)^2 = \text{ID}_A$$

In other words,  $w_A$  is equal to User A's ID,  $\text{ID}_A$ , as you can see in Figure 4-1.

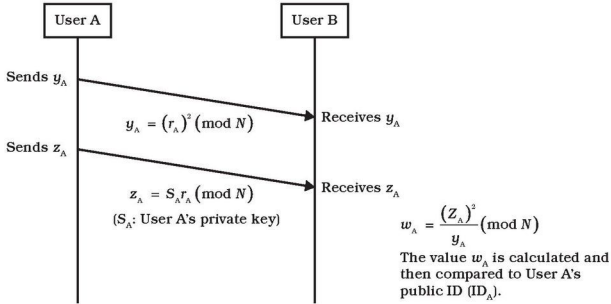


Figure 4-1: Verification process using a zero-knowledge interactive proof

## METHODS OF IDENTITY FRAUD

Building on the same example, let's consider how User X, an attacker, might try to steal User A's identity. If User A can verify themselves to User X while ensuring that there is only a very small probability User X can learn anything about User A's secret key, then the scheme is a valid zero-knowledge protocol. Otherwise, the protocol is faulty.

The zero-knowledge protocol requires that the prover (in this case, User A or User X) know the secret key to be able to prove their identity to the verifier (User B), so the protocol protects against identity attacks in which the adversary doesn't have knowledge of the secret key. User X, the adversary trying to identify as User A, doesn't know anything about User A's private key  $S_A$ . In an attempt to trick User B, User X selects two numbers  $e$  and  $f$  that have the following relationship with  $\text{ID}_A$ :

$$e^2 = \text{ID}_A \cdot f \pmod N$$

We'll discuss this equation in more detail later.

In the scenario from Figure 4-1, the legitimate User A first sends User B  $y_A$  and  $z_A$  to initiate a verification request. When attempting identity fraud, User X mimics this verification request by sending  $f$  and  $e$  to User B as though they were User A's credentials.

Let's use  $e = 25$  and  $f = 82$  to see how User X would try to mimic User A's ID ( $ID_A$ ).

Once User X sends  $e$  and  $f$  in place of  $z_A$  and  $y_A$ , respectively, User B uses  $e$  and  $f$  to calculate  $w_A$  as they would with a legitimate request. Because User X generated  $e$  and  $f$  to have a specific relationship with  $ID_A$ ,  $w_A$  will be identical to User A's public ID ( $ID_A = 101$ ), and User B will mistakenly authenticate User X as User A.

Let's see how User B's calculations work with  $e$  and  $f$ .

First, let's look at  $e$ 's relationship with  $v_A$ . First, we know that  $z_A = e$ , so we can substitute  $z_A$  for  $e$  in the following formula:

$$v_A = (z_A)^2 \pmod{N}$$

We find this relationship:

$$v_A = e^2 = 25^2 = 131 \pmod{247}$$

Next, we would substitute  $v_A = e^2$  and  $f = y_A$  into this equation:

$$w_A = \frac{v_A}{y_A} \pmod{N}$$

Doing so yields the following:

$$w_A = \frac{e^2}{f} = e^2 \oslash f^{-1} = v_A \oslash 82^{-1} \pmod{247}$$

Here, from  $82^{-1} \pmod{247} = 244$ , we get  $w_A = 131 \times 244 \pmod{247} = 101$ .

Using this method, User X can commit identity fraud on User A's public  $ID_A$  without knowing User A's private key  $S_A$  or the random number  $r_A$ . This process is illustrated in Figure 4-2.

To get around the verification process in Figure 4-1, User X needs to find values that fulfill the relationship  $f = e^2 \pmod{N}$  using only User A's public ID. To do this, User X squares  $e$  and divides by  $ID_A$  to find  $f$ :

$$f = \frac{e^2}{ID_A} \pmod{N}$$

Once User X calculates  $e$  and  $f$ , they can pretend to be User A without even knowing the random number  $r_A$ , committing a security breach. Let's look at a way to turn this protocol into one in which only User A, or someone with actual knowledge of the secret key  $S_A$ , can be verified as User A.

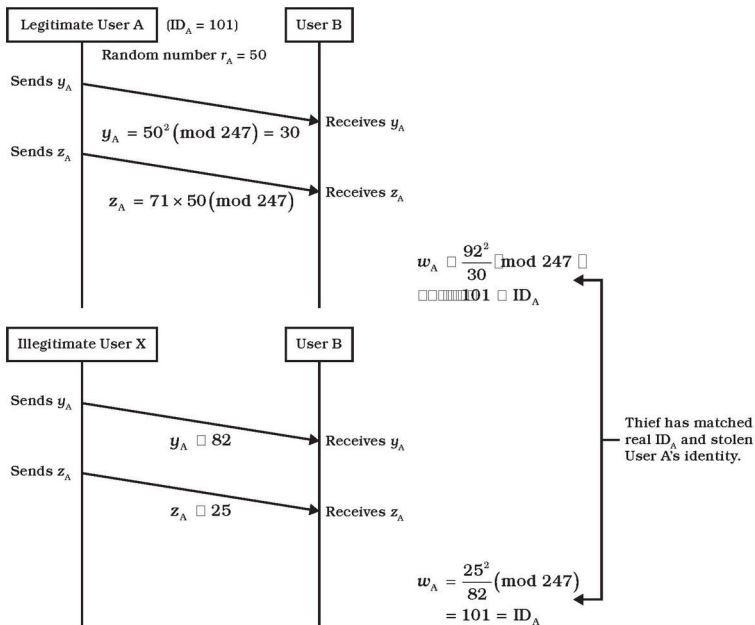


Figure 4-2: Example of identity theft

## ANTI-IDENTITY FRAUD IN A ZERO-KNOWLEDGE INTERACTIVE PROOF

Our protocol is flawed because identity fraud is possible, so let's look at how one can defend against an attack. To do this, we'll incorporate a randomly chosen *challenge bit*  $b$  and modify the prover and verifier's protocol, as shown in Figure 4-3.

When User A delivers  $y_A$  to User B, User B randomly chooses a value of 0 or 1 as the challenge bit  $b$  and sends  $b$  to User A. The prover then sends  $z_A$  to User B, where  $z_A = r_A \times (S_A^b)$ . User B performs a check on the returned value to confirm whether the sender is a legitimate user. This check requires calculating  $z_A^2$  using the two methods  $z_A^2 = r_A^2 \times (ID_A^b)$  and  $z_A \times z_A$  (where  $z_A$  is the value that User A sent). User B then compares the results of the two methods of calculating  $z_A^2$  to determine whether they are equal:  $z_A \times z_A = r_A^2 \times (ID_A^b)$ ? If they are equal, then the verifier accepts the prover's claim to be User A.

This entire protocol is run  $t$  times to ensure that the verifier trusts the prover, because if  $b = 0$  (which occurs 50 percent of the time if User B chooses the challenge bit randomly), the prover may simply return  $r_A$ . This will trick the verifier into accepting the answer even though the prover doesn't have knowledge

of  $S_A$ . The number of times the challenge bit check is performed depends on the level of secrecy one wants to achieve, where the probability a user will falsely impersonate someone is  $1/(2^t)$ . Performing the check  $t = \log(N)$  times isn't uncommon.

As long as User X doesn't have the secret key  $S_A$  and we run this experiment enough times to satisfy the verifier, this protocol can authenticate a user and address the risk of identity fraud. The verifier also doesn't learn any information about  $S_A$  from the protocol because  $z_A$  is calculated from  $S_A r_A \pmod{N}$ , which multiplies  $S_A$  by a random number.

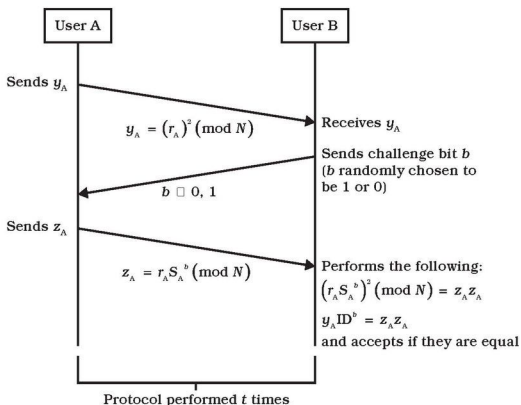


Figure 4-3: Valid verification process that prevents identity fraud

This protocol uses a small number of computations, so it's not uncommon to see this method on a token card of some sort.

# WHAT NEXT?

Though we've covered the main concepts of cryptography in this book, we've forgone exploring several important topics, including the future of cryptography. If you're curious about where cryptography will develop from here or what areas you could explore next, the following sections will give you an idea of where to start.

## PSEUDORANDOM NUMBERS AND ENCRYPTION SECURITY

A sequence of random numbers is one of the foundational pieces underlying security. For example, in public-key encryption, the key used to encrypt and decrypt information is generated using a random number. To ensure a good encryption scheme, one must use a random number generator whose results are impossible or very hard to predict (that is, one that generates a random number that can't be predicted based on previous numbers).

Pseudorandom numbers must be distinguished from the concept of *physically random numbers* (also called *truly random numbers*). Pseudorandom numbers are produced from a fixed formula (called a pseudorandom number generator), so they are deterministic or predictable by individuals with enough computing power. Some examples of pseudorandom number generators include the linear congruential generator, the middle square method, the M-series, the BBS (Blum-Blum-Shub) method, and the one-way hash function.

In contrast, a physically random number is generated based on a natural physical phenomenon, implements a perfectly random sequence of numbers, and can generate an infinite chaotic sequence of numbers. For example, a physical random number can be generated from the noise produced when electricity is distributed to a semiconductor circuit.

Despite the fact that most pseudorandom numbers are not secure, most systems use them instead of physically random numbers because many pseudorandom number generators require an inordinate amount of computing power to distinguish them from random number generators. In addition, the cost of using a truly random number generator is magnitudes higher than that of a pseudorandom number generator. We can't treat pseudorandom numbers as though they are truly random numbers, but in most cases, you can reasonably assume the adversary isn't capable of harnessing that much computing power, so pseudorandom generators suffice. Some top-secret or mission-critical systems may warrant the use of a random number generator, but these are rare.

## PGP

The initials *PGP* stand for Pretty Good Privacy. PGP is a widely used type of encryption software that was invented in 1991 by Philip Zimmermann.

PGP is a vital piece of functionality in nearly all modern encryption software. It makes certificate generation possible in symmetric-key algorithms (AES, 3-DES, and so on), public-key encryption (RSA, ElGamal, and so on), digital signatures (RSA, DSA), and one-way hash functions (MD5, SHA-1, RIPEMD-160, and so on).



## SSL/TLS

Communication protocols are used in online shopping and other forms of internet correspondence. When a protocol is used, a message authentication code certifies and authenticates the contents of a correspondence. Two security protocols that are often used for this purpose are *SSL (secure socket layer)* and *TLS (transport layer security)*. For example, when a credit card number or other secure information is sent through a web browser, SSL or TLS is used to encrypt the data and prevent it from being intercepted. To determine whether you are using SSL/TLS in online communication, you can inspect the URL in your internet browser. Non-SSL/TLS URLs begin with *http://*, while URLs beginning with *https://* indicate SSL/TLS communication.

Like SSL/TLS, the protocols called SMTP (Simple Mail Transfer Protocol) and POP3 (Post Office Protocol) also encrypt emails using SSL/TLS to protect information. SMTP is used to encrypt mail that is being sent out, while POP3 is used for received mail.

## QUANTUM CRYPTOGRAPHY

Quantum cryptography is considered an absolutely secure method of encryption. Quantum cryptography makes use of light pulses that contain more than 10,000 *photons*, which are the smallest unit of measurement for light. A single photon contains 1 bit of information, and the photon's polarity (the direction in which the electromagnetic waves oscillate) is used to distinguish whether the bit is a 0 or a 1. Photons follow *Heisenberg's uncertainty principle*, which states that a quantum system can't be observed without changing the system's state. This means the data contained in photons can't be copied by an attacker because, if a photon were observed by the attacker, the photon's polarization would be changed and any suspicious activity would be detected. This means that quantum encryption can't be intercepted. When quantum cryptography is combined with a one-time pad that uses a single-use encryption key, cryptographers can create fail-safe encryption.

## BIOMETRICS

Biometrics are biological features unique to an individual that are used to confirm a person's identity. Some examples of biometrics include fingerprints, palm veins, facial structures, iris patterns, palm prints, DNA (genes), and so on. Verification systems are seeing more everyday usage, one prominent example being the fingerprint unlock mechanism in smartphones.



IF WE XOR THE HINT ON PAGE 214:

00000010 00001100 00010101 00011100 00011101 00000110 00010010

AND THE WORD *LIBERTY* IN ASCII BINARY:

01101100 01101001 01100010 01100101 01110010 01110100 01111001

WE GET

01101110 01100101 01110111 01111001 01101111 01110010 01101011

WHICH IS *NEWYORK* IN ASCII BINARY.

# INDEX

## A

- addition (modulo operations), 137–138
- Advanced Encryption Standard (AES), 84–87
- algorithms, 20. *See also* symmetric-key algorithms
  - Euclidean algorithm, 181–182
  - exhaustive search algorithm, 81
  - extended Euclidean algorithm, 181–183
- anti-identity fraud, 222–223
- associative property (modulo operations), 149
- asymmetric cryptography, 110–113. *See also* public-key encryption
- authentication, 114, 217. *See also* message authentication codes

## B

- BBS (Blum-Blum-Shub) method, 224
- binary digits, 48–58
  - ASCII, 54
  - bits, 53
  - byte, 53
  - converting plaintext into binary, 89
  - hexadecimal notation, 53
  - logical operators and
    - bitwise operation, 56
    - AND operations, 56
    - OR operations, 56
    - XOR operation, 55–58

- biometrics, 225
- bits, 53
  - origin of term, 19
  - reduced DES
    - compressing, 101–104
    - higher-order bits, 92, 95, 97, 99
    - lower-order bits, 92, 95, 97, 99
    - rotating, 102, 103

- block ciphers, 65
  - CBC mode, 71
  - ECB mode, 70
  - hexadecimal notation, 68–69
  - key size, 68–69
  - methods for breaking, 81
  - padding, 69
- Blum-Blum-Shub (BBS) method, 224
- brute force attacks, 67

## C

- Caesar cipher, 24
- CBC (cipher block chaining) mode, 71
- certificate authority, 204
- certificates (server certificates), 202, 204, 209–210, 213
- CFB (cipher feedback), 71
- character encoding, 38
  - ASCII, 54
  - in hexadecimal notation, 68–69
  - RSA encryption, 169, 172
- checksum process, 193
- cipher block chaining (CBC) mode, 71
- cipher feedback (CFB), 71

## ciphers, 17

- block ciphers, 65
  - CBC mode, 71
  - ECB mode, 70
  - hexadecimal notation, 68–69
  - key size, 68–69
  - methods for breaking, 81
  - padding, 69
- Caesar cipher, 24
- codes versus, 18
- computationally secure, 39
- DES ciphers
  - decryption process, 79
  - disadvantages of, 81
  - encryption process, 79
  - Feistel ciphers, 73–74, 92–95
  - involution, 74
  - key size, 75
  - NSA standards, 75–76
  - parity check, 75
  - reduced DES versus, 104
  - round function  $f$ , 78
  - subkeys, 77
- encryption model, 19–23
- fail-safe ciphers, 39
- polyalphabetic cipher, 26
- reduced DES
  - converting data into binary, 89
  - decryption process, 97–100
  - DES versus, 104
  - encryption process, 90–96
  - key schedule, 90, 100–104

- reduced DES (*continued*)
    - relationship between
      - encryption key and decryption key, 21–23
    - security, 28–46
      - breaking classical
        - encryption, 37–40
      - combinations, 33
      - frequency analysis, 33
      - permutations, 32
      - polyalphabetic cipher's key
        - space, 33–34
      - secure ciphers, 39–40
      - substitution cipher's
        - key space, 31–32
      - transposition
        - cipher's key space, 34–37
    - stream ciphers, 65–67
    - substitution cipher, 25, 31–32
    - Tanuki cipher, 6
    - transposition cipher, 27, 34–37
    - Vernam cipher, 37–39
  - ciphertext, 20. *See also*
    - names of specific encryption methods*
  - breaking classic encryption, 37–40
  - Caesar cipher, 24
  - polyalphabetic cipher, 26
  - simple substitution cipher, 25
  - substitution cipher, 25
  - transposition cipher, 27
  - RSA encryption
    - decrypting ciphertext, 171–172
    - generating ciphertext, 169–171
  - classic encryption
    - breaking, 37–40
      - perfectly secure encryption, 37–38
    - secure ciphers, 39–40
    - Caesar cipher, 24
    - polyalphabetic cipher, 26
    - simple substitution cipher, 25
    - substitution cipher, 25
    - transposition cipher, 27
    - codes (jargon: procedure words), 18
    - collision attacks, 193
    - combinations, 33
    - commutative property (modulo operations), 149
    - composite numbers
      - creating, 217
      - prime numbers and, 123–124
    - computationally infeasible problem, 33
    - computationally secure ciphers, 39
    - contraposition, 154
    - conversion rule  $\delta$  (delta), 26
    - conversion rule  $\sigma$  (sigma), 25
    - coprime numbers, 143
      - Euler's totient function, 156
    - primality tests, 155
  - credit cards
    - SSL/TLS encryption, 225
    - zero-knowledge interactive proof, 217–218
  - cryptography. *See also*
    - public-key encryption; symmetric-key algorithms
    - asymmetric cryptography, 110–113
    - importance of, 11–13
    - quantum cryptography, 225
  - currency (money), 207
- D**
- Daemen, Joan, 85
  - Data Encryption Standard ciphers. *See* DES ciphers
  - decryption key
    - generating, 79, 101, 165–166
    - relationship between encryption key and, 21–23
  - decryption process, 20
    - DES ciphers, 79
    - reduced DES
      - expansion function, 97–98
      - final permutation, 99
      - initial permutation, 97
      - relationship between encryption process and, 100
      - separating data into higher-order and lower-order bits, 97
      - swapping higher-order and lower-order bits, 99
    - RSA, 163–164, 171–172
  - delta ( $\delta$ ) conversion rule, 26
  - DES (Data Encryption Standard) ciphers, 72
    - decryption process, 79
    - disadvantages of, 81
    - encryption process, 79
    - Feistel ciphers, 73–74, 92–95
    - involution, 74
    - key size, 75
    - NSA standards, 75–76
    - parity check, 75
    - reduced DES
      - converting data into binary, 89
    - decryption process, 97–100

- DES versus, 104
  - encryption process, 90–96
  - key schedule, 90, 100–104
  - reduced DES versus, 104
  - round function  $f$ , 78
  - subkeys, 77
  - differential cryptanalysis, 81
  - Diffie–Hellman key exchange method, 178
  - digital signatures, 200–202
    - certificates and, 204–205
    - digital signature scheme, 179
    - man-in-the-middle attacks, 203–205
    - PGP and, 224
  - discrete logarithm problem, 115, 173–175
  - ElGamal encryption, 176–180
  - extended Euclidean algorithm, 181–183
  - one-way functions, 116
  - distributive property (modulo operations), 149
  - division (modulo operations), 143–147
  - divisors (factors), 122
- E**
- electric code book (ECB) mode, 70
  - ElGamal encryption (discrete logarithm problem), 176–180
  - Encrypt-Decrypt-Encrypt (EDE) mode, 83
  - encryption, 20. *See also* public-key encryption; symmetric-key algorithms
  - ciphers, 17
    - block ciphers, 68–71
    - Caesar cipher, 24
    - classic encryption, 24–27, 37–40
    - codes versus, 18
    - combinations, 33
    - DES ciphers, 72–81
    - encryption model, 19–23
    - frequency analysis, 33
    - permutations, 32
    - polyalphabetic cipher, 26, 33–34
    - relationship between encryption key and decryption key, 21–23
    - secure, 39–40
    - security, 28–46
    - stream ciphers, 65–67
    - substitution cipher, 25, 31–32
    - transposition cipher, 27, 34–37
  - digital signatures, 200–205
  - ElGamal encryption, 176–180
  - hash functions, 190–194
  - hybrid encryption, 185–189
  - message authentication codes, 195–199
  - public-key infrastructure, 206–214
  - RSA encryption, 161–164
    - decrypting ciphertext, 171–172
    - generating ciphertext, 169–171
    - origin of, 119
    - private-key generation, 165–168
    - public-key generation, 165–168
  - SSL/TLS encryption, 189, 197, 225
  - zero-knowledge interactive proof, 217–223
  - encryption key, 20
    - generating, 79, 101, 165–166
    - relationship between decryption key and, 21–23
  - encryption process
    - DES ciphers, 79
    - reduced DES, 90–91
      - Feistel network, 92–95
      - final permutation, 95–96
      - initial permutation, 91
      - relationship between decryption process and, 100
    - separating data into higher-order and lower-order bits, 92
    - swapping higher-order and lower-order bits, 95
  - Shannon's encryption model, 19–23
  - Eratosthenes of Cyrene, 125
  - Euclidean algorithm, 181–182
  - Euler, Leonhard, 157
  - Euler's formula, 157
  - Euler's totient function (theorem), 156–160
  - exhaustive search algorithm, 81
  - extended Euclidean algorithm (discrete logarithm problem), 181–183
- F**
- factors (divisors), 122
  - fail-safe ciphers, 39
  - falsification, 189
  - Federal Information Processing Standard (FIPS), 85
  - Feistel, Horst, 72
  - Feistel ciphers (Feistel network), 73–74, 92–95



Fermat, Pierre de, 153  
 Fermat's big (last) theorem, 153  
 Fermat's little theorem (Fermat's method), 152–155  
 Ferris wheel model (modulo operations)  
   addition and subtraction, 136–139  
   multiplication and division, 146–147  
 fields (modulo operations), 149  
 final permutation (FP), 95–96, 99  
 finite fields, 149  
 FIPS (Federal Information Processing Standard), 85  
 FP (final permutation), 95–96, 99  
 fractions, 121  
 frequency analysis, 33

## G

*The Gold Bug* (Poe), 35–36

## H

hash functions, 190–194, 224  
 Heisenberg's uncertainty principle, 225  
 higher-order bits (reduced DES decryption process)  
   separating data into, 92, 97  
   swapping with lower-order bits, 95, 99  
 hybrid encryption, 185–186  
   public-key encryption process, 187–189  
   symmetric-key algorithms, 187

## I

identity fraud  
   countermeasures, 195–199  
   zero-knowledge interactive proof, 220–223

implementation stage (zero-knowledge interactive proof), 218–220  
 infinite fields (modulo operations), 149  
 inflation, 207  
 initialization vector, 71  
 initial permutation (IP), 91, 97  
 integer factorization problem, 115  
   Euler's Theorem, 156–160  
   Fermat's little theorem, 152–155  
   modulo operations, 134–137  
     addition, 137–138  
     associative property, 149  
     commutative property, 149  
     distributive property, 149  
     division, 143–147  
     fields, 149  
     multiplication, 140–143, 146–147  
     power table, 150–151  
     subtraction, 139  
 one-way functions, 116  
 prime numbers and, 120–123  
   composite numbers and, 123–124  
   primality tests, 129–133  
   sieve of Eratosthenes, 124–129  
 RSA encryption, 119, 161–172  
 integers, 121  
 integrity (messages), 193, 197  
 inverse element (reciprocal), 144–145  
 involution (DES ciphers), 74  
 IP (initial permutation), 91, 97

irrational numbers, 121  
 irreversibility property, 193

## J

jargon (codes), 18

## K

key schedule (reduced DES), 90, 100–101  
   compressing bits, 103–104  
   PC-1, 101–102  
   PC-2, 102–103  
   rotating bits, 102, 103  
 key size  
   DES ciphers, 75  
   3-DES, 83–84  
 key space, 29  
   polyalphabetic cipher, 33–34  
   substitution cipher, 31–32  
   transposition cipher, 34–37

## L

letter-by-letter replacement method, 25  
 linear congruential generator, 224  
 linear cryptanalysis, 81  
 lower-order bits  
   reduced DES decryption process  
     separating data into, 97  
     swapping with higher-order bits, 99  
   reduced DES encryption process  
     separating data into, 92  
     swapping with higher-order bits, 95  
 lowest common multiple, 158

## M

MACs (message authentication codes), 195–199  
man-in-the-middle attacks, 203–205  
“A Mathematical Theory of Communication” article (Shannon), 19  
MD5 hash function, 193  
message authentication codes (MACs), 195–199  
middle square method, 224  
Miller-Rabin method (primality tests), 129  
modulo operations, 134–137  
    addition, 137–138  
    associative property, 149  
    commutative property, 149  
    distributive property, 149  
    division, 143–147  
    fields, 149  
    multiplication, 140–143, 146–147  
    power table, 150–151  
    subtraction, 139  
M-series pseudorandom number generator, 224  
multiplication (modulo operations), 140–143, 146–147

## N

National Security Agency (NSA) standards (DES ciphers), 75–76  
natural numbers, 121, 181–182  
nonnegative integers, 121

## O

OFB (output feedback), 71  
one-time pad, 37, 225  
one-way functions, 115–118  
    discrete logarithm problem, 116

integer factorization problem, 115  
output feedback (OFB), 71

## P

padding (block ciphers), 69  
parity check (DES ciphers), 75  
PC-1 (permuted choice 1) (DES keys), 77, 101–102  
PC-2 (permuted choice 2) (DES keys), 77, 102–103  
perfectly secure encryption, 37–38  
permutations, 32  
    final permutation, 95–96, 99  
    initial permutation, 91, 97  
permuted choice 1 (PC-1) (DES keys), 77, 101–102  
permuted choice 2 (PC-2) (DES keys), 77, 102–103  
PGP (Pretty Good Privacy), 189, 224  
photons (quantum cryptography), 225  
physically random numbers (truly random numbers), 224  
PKI (public-key infrastructure), 206–214  
plaintext, 20. *See also names of specific encryption methods*  
    breaking classic encryption, 37–40  
    Caesar cipher, 24  
    converting into binary, 89  
    polyalphabetic cipher, 26  
    simple substitution cipher, 25  
    substitution cipher, 25  
    transposition cipher, 27  
Poe, Edgar Allen, 35–36  
polyalphabetic cipher, 26, 33–34

POP3 (Post Office Protocol), 225  
power table (modulo operations), 150–151  
preparatory stage (zero-knowledge interactive proof), 217–218  
Pretty Good Privacy (PGP), 189, 224  
primality tests, 129–133  
prime numbers, 120–123  
    composite numbers and, 123–124  
    coprime numbers, 143  
        Euler's totient function, 156  
        primality tests, 155  
    primality tests, 129–133  
    sieve of Eratosthenes, 124–129  
primitive roots, 174  
private key  
    calculating and sending to user, 218  
    making, 167–168  
    RSA encryption  
        calculating, 182–183  
        generating, 165–168  
        verifying compatibility, 166  
procedure words (codes), 18  
proposition, 154  
provers (zero-knowledge interactive proof), 218–219  
pseudoprimes, 129, 155  
pseudorandom number generator, 224  
pseudorandom numbers, 66–67, 151, 155, 224  
public key  
    making, 167–168  
    RSA encryption  
        generating, 165–168  
        verifying compatibility, 166

- public-key encryption, 64, 106–109
- asymmetric cryptography, 110–113
- authentication, 114
- discrete logarithm problem, 115, 173–175
  - ElGamal encryption, 176–180
  - extended Euclidean algorithm, 181–183
- hybrid encryption, 187–189
- integer factorization, 115
  - Euler's Theorem, 156–160
  - Fermat's Little Theorem, 152–155
  - modulo operations, 134–151
  - prime numbers and, 120–133
  - RSA encryption, 119, 161–172
- one-way functions, 115–118
- PGP and, 224
- viewing public key, 109
- public-key infrastructure (PKI), 206–214

## Q

- quantum cryptography, 225

## R

- rational numbers, 121, 149
- reciprocal (inverse element), 144–145
- reduced DES
  - converting data into binary, 89
- decryption process
  - expansion function, 97–98
  - final permutation, 99
  - initial permutation, 97

- relationship between encryption process and, 100
- separating data into higher-order and lower-order bits, 97
- swapping higher-order and lower-order bits, 99

- DES versus, 104
- encryption process, 90–91

- Feistel network, 92–95
- final permutation, 95–96
- initial permutation, 91
- relationship between decryption process and, 100
- separating data into higher-order and lower-order bits, 92

- swapping higher-order and lower-order bits, 95

- key schedule, 90, 100–101

- compressing bits, 103–104
- PC-1, 101–102
- PC-2, 102–103
- rotating bits, 102, 103

- repository, 204, 209–210
- repudiation, 197, 199–200
- Rijmen, Vincent, 85
- Rijndael algorithm (Advanced Encryption Standard), 84–87

- round function  $f$  (DES ciphers), 78

- RSA encryption, 161–164
  - calculating private key generation, 182–183
  - decrypting ciphertext, 171–172

- generating ciphertext, 169–171
- origin of, 119
- private key generation, 165–168
- public key generation, 165–168

## S

- S-box (substitution box), 78, 93
- secret-key cryptography. *See* symmetric-key algorithms
- secure ciphers, 39–40
- secure socket layer/transport layer security (SSL/TLS) encryption, 189, 197, 225
- server certificates (certificates), 202, 204, 209–210, 213
- SHA-1 hash function, 193
- Shannon, Claude, 19, 37
- Shannon's encryption model, 19–23
- shared-key cryptography. *See* symmetric-key algorithms
- shared keys, 59–64
- sieve of Eratosthenes, 124–129
- sigma ( $\sigma$ ) conversion rule, 25
- simple substitution ciphers, 25
- SMTP (Simple Mail Transfer Protocol) protocol, 225
- SPN (substitution-permutation network), 85
- SSL/TLS (secure socket layer/transport layer security) encryption, 189, 197, 225
- stream ciphers, 65–67
- strong collision resistance (hash functions), 193
- subkeys (DES ciphers), 77

- substitution box (S-box), 78, 93
  - substitution cipher, 25, 31–32
  - substitution-permutation network (SPN), 85
  - substitution rule  $\tau$  (tau), 27
  - subtraction (modulo operations), 139
  - symmetric-key algorithms
    - AES, 84–87
    - binary digits, 48–58
    - block ciphers, 65
      - CBC mode, 71
      - ECB mode, 70
      - hexadecimal notation, 68–69
      - key size, 68–69
      - methods for breaking, 81
      - padding, 69
    - CBC mode, 71
    - DES ciphers, 72
      - decryption process, 79
      - disadvantages of, 81
      - encryption process, 79
    - Feistel ciphers, 73–74, 92–95
    - involution, 74
    - key size, 75
    - NSA standards, 75–76
    - parity check, 75
    - round function  $f$ , 78
    - subkeys, 77
  - hybrid encryption, 187
  - logical operators, 55–58
  - PGP and, 224
  - reduced DES
    - converting data into binary, 89
    - decryption process, 97–100
    - DES versus, 104
    - encryption process, 90–96
    - generating keys, 100–104
    - shared keys, 59–64
    - stream ciphers, 65–67
    - 3-DES, 82
      - EDE mode, 83
      - key size, 83–84
  - symmetric-key scheme, 112
- T**
- tampering countermeasures, 190–194
  - Tanuki cipher, 6
  - tau ( $\tau$ ) substitution rule, 27
  - 3-DES, 82
    - EDE mode, 83
    - key size, 83–84
  - transposition cipher, 27, 34–37
  - trapdoor functions, 115–118
    - discrete logarithm problem, 116
    - integer factorization problem, 115
  - triple-DES. *See* 3-DES
  - truly random numbers (physically random numbers), 224
- U**
- unique arrangements (ciphers), 32
  - unique factorization theorem, 124
- V**
- verification, 114
  - verifiers (zero-knowledge interactive proof)
    - calculating and sending private key to user, 218
    - creating composite number, 217
    - registering user ID with, 217
  - Vernam, Gilbert, 37
  - Vernam cipher, 37–39
- W**
- Wiles, Andrew, 153
- Z**
- zero-knowledge interactive proof, 217
    - anti-identity fraud, 222–223
    - identity fraud, 220–222
    - implementation stage, 218–220
    - preparatory stage, 217–218

# ABOUT THE AUTHORS

## MASAAKI MITANI

Masaaki Mitani is from the town of Setoda, formerly known as Toyota, in Onomachi of the Hiroshima prefecture. He graduated in 1974 from the Department of Electrical and Electronic Engineering in the School of Engineering at the Tokyo Institute of Technology. After working as an assistant professor at his alma mater, he is now a professor at Tokyo Denki University's School of Engineering in the Department of Information and Communication Engineering. He specializes in digital signal processing engineering, communication engineering, and educational engineering. He has a PhD in Engineering.

### *PRINCIPAL WORKS*

- *An Introduction to Digital Signal Processing* (Ohmsha, Ltd.)
- *Learning Digital Signal Processing using Scilab* (CQ Publishing)
- *A Friendly Introduction to Electronic Circuits, Introduction (I) to Diodes and Transistors* (Micronet)
- *Re-Learning Signal Mathematics* (CQ Publishing)
- *Immediately Applicable Fourier Transform* (Kodansha Ltd.)

## SHINICHI SATO

Shinichi Sato is from the town of Date in the Fukushima prefecture. In 1990, he earned his masters degree in electrical engineering at Tokyo Denki University's graduate student program. He worked on designing imaging machinery at a private corporation before conducting research related to bionics in a private institution's medical faculty as an assistant researcher. He is now an assistant professor at Tokyo Denki University's School of Engineering in the Department of Information and Communication Engineering. He specializes in signal processing engineering and educational engineering.



# PRODUCTION TEAM FOR THE JAPANESE EDITION

**Production:** Verte Corp.

**Editing:** Tsugumi Endo

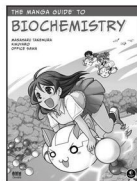
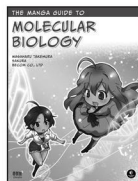
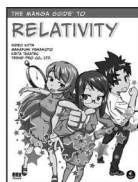
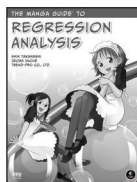
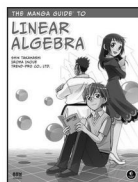
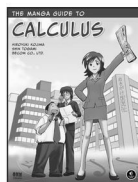
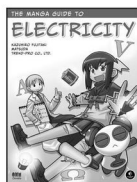
**DTP:** Satoshi Arai

## HOW THIS BOOK WAS MADE

The *Manga Guide* series is a co-publication of No Starch Press and Ohmsha, Ltd. of Tokyo, Japan, one of Japan's oldest and most respected scientific and technical book publishers. Each title in the best-selling *Manga Guide* series is the product of the combined work of a manga illustrator, scenario writer, and expert scientist or mathematician. Once each title is translated into English, we rewrite and edit the translation as necessary and have an expert review each volume. The result is the English version you hold in your hands.

# MORE MANGA GUIDES

Find more *Manga Guides* at your favorite bookstore, and learn more about the series at <https://www.nostarch.com/manga/>.





# DECODE THE MYSTERIES OF CRYPTOGRAPHY



INSPECTOR JUN MEGURO HAS BEEN TASKED WITH TRACKING DOWN A CRAFTY ART THIEF WHO LEAVES CLUES IN THE FORM OF PERPLEXING CRYPTOGRAPHIC PUZZLES. LUCKY FOR HIM, HIS SISTER RUKO IS A MATH WHIZ WHO IS WILLING TO TEACH HIM ABOUT CRYPTOGRAPHY SO THAT HE CAN CRACK THE CASE. BUT CAN THEY DECODE THE THIEF'S CIPHERS BEFORE SHE STEALS AGAIN?

AS YOU FOLLOW JUN AND RUKO'S QUEST TO BRING THE CIPHER-WIELDING THIEF TO JUSTICE, YOU'LL LEARN REAL CRYPTOGRAPHY AND THE MATH BEHIND IT, INCLUDING:

THE FOUNDATIONS OF ENCRYPTION, LIKE THE CAESAR AND VERNAM CIPHERS

HOW SUBSTITUTION CIPHERS, PUBLIC-KEY CRYPTOGRAPHY, AND DIGITAL SIGNATURES WORK

HOW TO USE BINARY, LOGIC OPERATIONS, AND MODULAR ARITHMETIC TO TURN PLAINTEXT MESSAGES INTO NUMERIC VALUES THAT CAN BE ENCRYPTED

HOW THE DATA ENCRYPTION STANDARD (DES) CIPHER WORKS

HOW TO ENCRYPT MESSAGES WITH HASH FUNCTIONS, MESSAGE AUTHENTICATION CODES (MACs), AND PUBLIC KEY INFRASTRUCTURES (PKIs)

**THE MANGA GUIDE TO CRYPTOGRAPHY** IS THE PERFECT INTRODUCTION TO CRYPTOGRAPHY FOR PROGRAMMERS, SECURITY PROFESSIONALS, AND ASPIRING CRYPTOGRAPHERS.

FIND MORE MANGA GUIDES AT [WWW.NOSTARCH.COM/MANGA](http://WWW.NOSTARCH.COM/MANGA)

SHELVES IN: COMPUTERS/SECURITY



THE FINEST IN GEEK ENTERTAINMENT™  
[www.nostarch.com](http://www.nostarch.com)