SPECIAL REPORT

Computer Security and the Internet

How Hackers Break In... and How They Are Caught

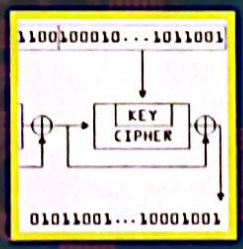
by Carolyn P. Meinel

How Computer Security Works

1 Firewalls by William Cheswick & Steven M. Bellovin
2 Digital Certificates by Warwick Ford
3 The Java Sandbox by James Gosling

Cryptography for the Internet

by Philip R. Zimmermann



The Case against Regulating Encryption Technology

by Ronald L. Rivest

This past February hackers reached through the Internet to break into the computer networks at various U.S. Air Force and Navy sites. The intruders, allegedly two northern Californian teenagers, were trying to gain access to systems that contained sensitive shipping, personnel and payroll information. At one point, the Pentagon thought that Saddam Hussein might be responsible for the attacks—a suspicion that could have had disastrous consequences. Such electronic transgressions are hardly rare: a recent study co-authored by the Federal Bureau of Investigation found that nearly two thirds of the organizations and companies surveyed were the victims of a cyberviolation within the past year.

In this special report, experts describe advanced technologies for thwarting cybertrespassers, thieves and eavesdroppers. Defensive software and hardware, such as firewall servers, can detect and block intruders; advanced encryption techniques ensure the privacy of data should a security breach occur. Cryptography also enables confidential messages to be dispatched freely around the world over the open Internet. Such approaches for securing computers—and the electronic data they store, transmit and receive—will help preserve the Net as a shared, invaluable tool.

—The Editors

How Hackers Break In...

Port scanners, core dumps and buffer overflows are but a few of the many weapons in every sophisticated hacker's arsenal. Still, no hacker is invincible

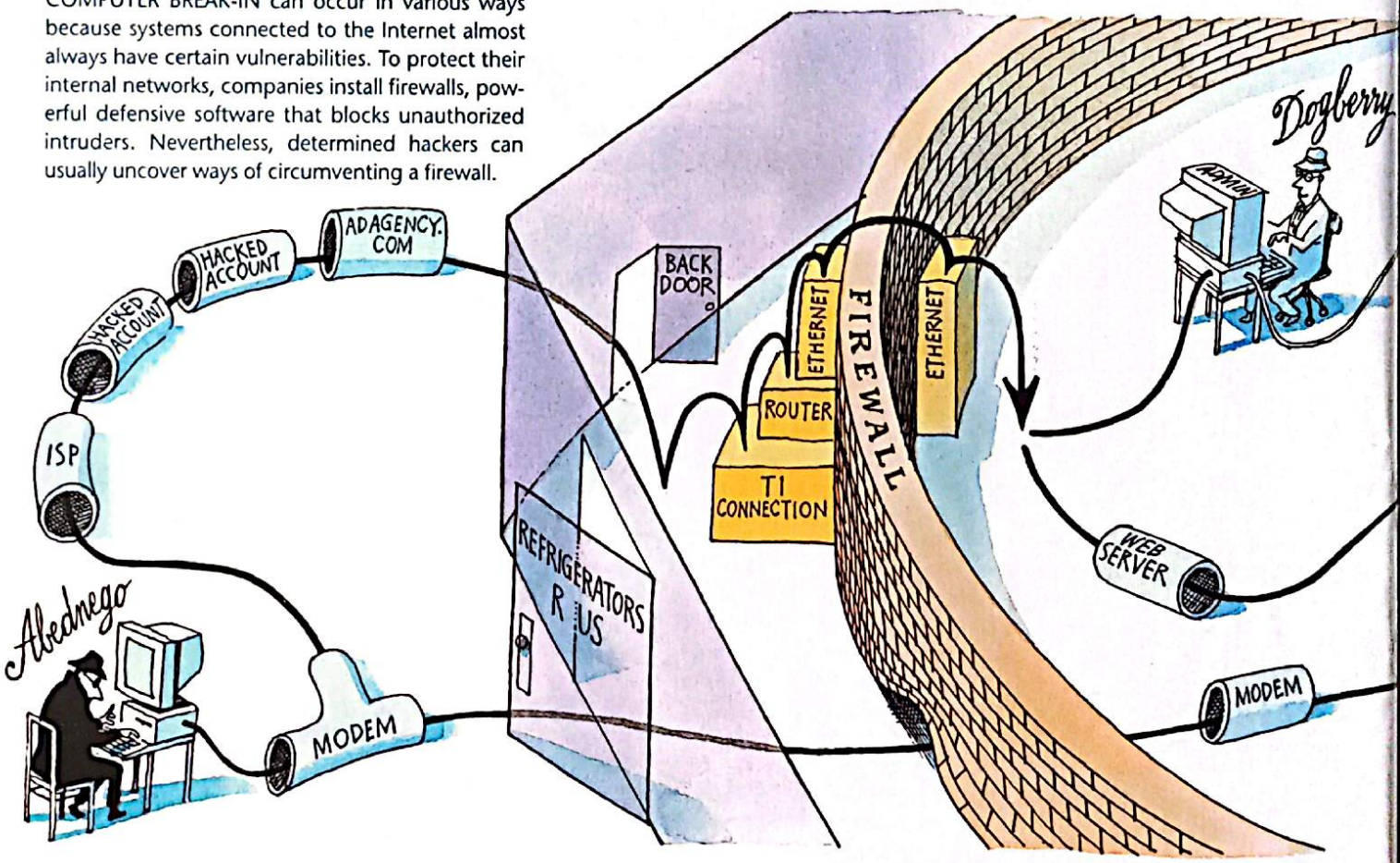
by Carolyn P. Meinel

Editors' note: *This fictionalized account is a composite of many incidents that have occurred, at one time or another, somewhere in cyberspace. The names of people and other details have been changed, but the technologies and software do exist. Some of the events reported here are drawn from the firsthand experiences of the author, who is known both in the computer underground and among security experts for her hacking skills and for her countless battles against hackers.* SCIENTIFIC AMERICAN *thanks Rt66 Internet, an Internet service provider in Albuquerque, N.M., which tested much of the software and hardware described in this article to verify the technologies involved.*

Sitting at his home computer one night, Abednego logs on to the Internet Relay Chat, the cyberspace equivalent of CB radio. After connecting to a channel devoted to the powerful Unix operating system, he watches as the on-line habitués meet to make contacts, build alliances and exchange knowledge. The scene is reminiscent of the cantina in *Star Wars*.

Eager to interject himself into the conversation—and to impress others—Abednego waits for someone to ask a simple-minded question so that he can incite a "flame war," in which the participants begin hurling venomous insults at

COMPUTER BREAK-IN can occur in various ways because systems connected to the Internet almost always have certain vulnerabilities. To protect their internal networks, companies install firewalls, powerful defensive software that blocks unauthorized intruders. Nevertheless, determined hackers can usually uncover ways of circumventing a firewall.



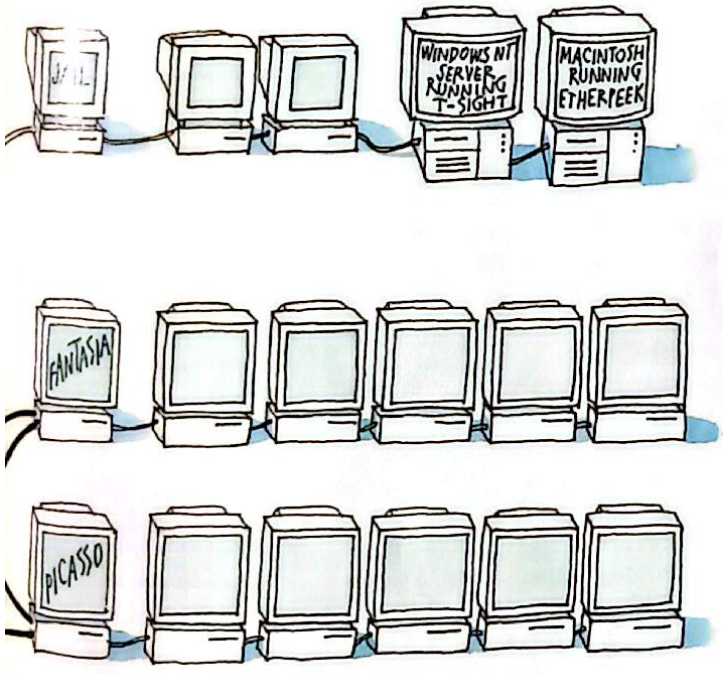
and How They Are Caught

one another. Just then, someone with the handle "Dogberry" asks about writing a device driver for a home weather station. Abednego seizes his chance. "RTFM" is his response. It stands for "read the f——g manual."

Others begin launching nasty insults, but not at Dogberry. Apparently, the question was far more complex than Abednego had realized. Dogberry's terse put-down—"Newbie!"—fans the flames. Humiliated, Abednego vows revenge.

Using the "finger" command on Internet Relay Chat, Abednego obtains the e-mail address "Dogberry@refrigerus.com." Abednego figures that if Dogberry is such a Unix whiz, he might be manager of the computers at refrigerus.com. To confirm his hunch, Abednego uses "telnet" to connect to the mail server of that computer. He then issues the command "expn root@refrigerus.com" and learns that Dogberry is indeed the head system administrator there.

His interest sufficiently piqued, Abednego runs Strobe, a program that attempts to connect with each of the thousands of virtual ports on refrigerus.com. The scanner will meticulously record responses from any daemons, which are automatic utility programs, such as those that handle e-mail. Abednego knows that each port might be an open door—or a door that he might be able to break down—if he can take advantage of some flaw in its daemon.



ALL ILLUSTRATIONS BY DUSAN PETRICIC

But Strobe hits a wall—Dogberry's firewall, to be exact. That powerful defensive software intercepts each incoming packet of data, reads its TCP/IP (transmission control protocol/Internet Protocol) header and determines with which port it seeks to connect. The firewall compares this request with its own strict rules of access. In this case, refrigerus.com has decreed that there should be only one response to Abednego's scanner.

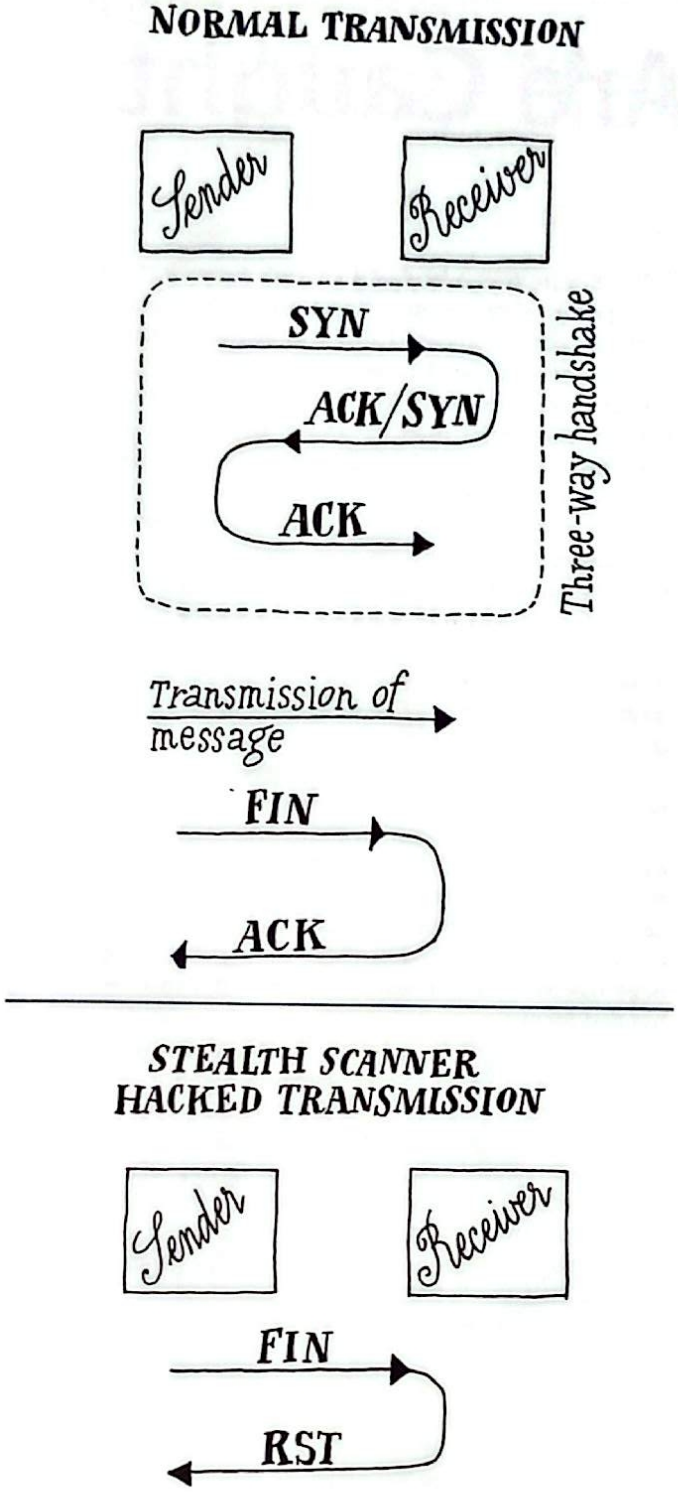
From that instant on, a program on refrigerus.com sends a blitzkrieg of meaningless data, including random alphanumeric characters, back to Abednego, overwhelming his home PC. Meanwhile another daemon sends e-mail to Abednego's Internet service provider (ISP), complaining that someone is attempting to break into refrigerus.com. Within minutes, the ISP closes Abednego's account for suspicion of computer crime.

Although Abednego is caught off guard—many ISPs would not have taken such a strong measure so quickly—the setback is minor. The closed account was only one of several he had created after breaking into that ISP. But the termination of the account at that particular moment causes him to be dumped from Internet Relay Chat in the midst of the flames against him. To the others on-line, it looks as if Abednego has been unceremoniously booted or, worse, that he has fled for cover.

Abednego burns for retaliation. His next step is to try a stealth port scanner. Such programs exploit the way in which IP transmissions work. When one computer wishes to talk to another, it must first transmit a short message packet containing a SYN (synchronize) flag. The header of the packet also contains other important information, such as the IP address of both the source and destination. In response, the recipient daemon sends back a packet that contains an ACK (to acknowledge the received packet), a SYN and a sequence number that is used to coordinate the upcoming transmission. When the first computer gets the return ACK/SYN, it issues an ACK of its own to confirm that all is ready, thus completing a three-way handshake. Then, and only then, can the sender computer begin transmitting its message using the sequence number provided. At the end of the communication, the sender transmits a packet with a FIN (finish) flag, and the receiver returns an ACK to signal that it is aware the transmission has ended.

Abednego knows that a stealth port scanner can take advantage of this process by sending just premature FIN packets to each port on a computer. Typically if a port is open, the recipient daemon will not send any response. If a port is closed, however, the computer will return an RST (reset) packet. But because this computer does not truly recognize a connection until it has completed the opening three-way handshake, it does not record the transmission in its logs. Thus, a FIN scanner can probe a computer in relative secrecy, without ever having opened any official connections. (Yet, as Abednego will soon learn, there is enough information in even one FIN packet to establish a sender's identity.)

Abednego surfs the Internet to search for an advanced



INTERNET TRANSMISSIONS follow certain rigid protocols. Normally, the sender first transmits an introductory message packet containing a SYN flag to synchronize the upcoming communication (*top*). The receiver then returns an ACK, which acknowledges the request, and a SYN. After obtaining this information, the sender transmits an ACK, which completes the necessary three-way handshake. Only then can the sender dispatch the message itself. When finished, he issues a FIN flag, and the receiver returns an ACK, which officially closes the correspondence. A hacker can circumvent the process by sending just a premature FIN, from which the hapless receiver might return an RST, or reset, packet (*bottom*). The response—or lack of one—reveals certain information about the receiver, but because no three-way handshake was ever completed, the transmission is not recorded in the receiver's logs. The hacker can thus probe an unwitting computer in relative secrecy.

stealth port scanner and finds one at an underground Web site. The program, like most other hacker tools, is written in the C computer language. Abednego struggles to compile, or convert, the scanner from C into a form that can be executed on his home PC, which runs on Linux, one of the many variants of Unix.

Abednego's difficulty in converting the software is not unusual because of the many peculiarities of the different flavors of Unix. And Abednego, like many hackers, did not formally study computer science. In fact, also like most hackers, Abednego never learned to program because he never had to: almost any software a computer criminal might ever want is available on the Internet, already written and free for the taking—as long as the hacker knows how to compile it (or has cohorts who do).

The young Dogberry had taken a different path. After befriending a technician at a local ISP, he learned how to administer a network. Before long, Dogberry and the technician were playing computer break-in and defense games. The payoff came when they used the results to help the ISP improve its security. With that success, Dogberry was hired by the ISP to work part-time while he pursued his computer science degrees.

Thus, when Abednego decided to take on Dogberry, he had already made his first mistake. Dogberry is a white-hat (or nonmalicious) hacker and a veteran of many cyberbattles.

Casing the Joint

As dawn breaks, Abednego has finally finished compiling the code and is ready to deploy it. Within minutes, the FIN scanner has given him a snapshot of the services that refrigerus.com offers to those coming only from an approved IP address. Two that draw his attention are a secure-shell daemon, which is a way to make encrypted Internet connections, and a Web server.

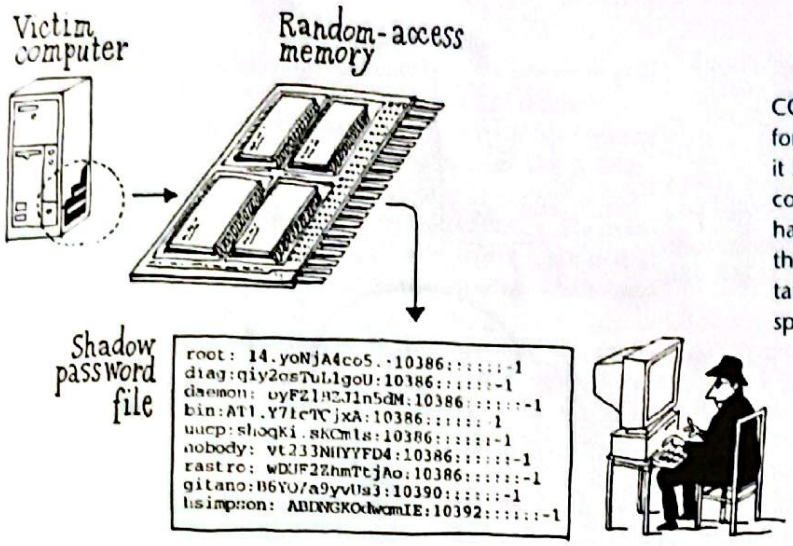
Then Abednego's heart skips a beat. An unusual port number, 31,659, has also turned up on his FIN scan. Could another intruder have preceded him and left a back door, a secret passage to enter the system undetected?

The beeping of a pager jolts Dogberry out of a deep sleep. EtherPeek, a sniffer program installed on the refrigerus.com network, has detected the port scan. Dogberry rushes into the office to watch for more attacks from the console of his administrative computer. His best defensive programs run only from that machine and only for someone who is physically there, so that they cannot be tampered with remotely by an attacker.

Meanwhile, despite the powerful temptation of that 31,659 daemon, Abednego leaves the chase for now. Something—his hacker intuition—tells him that he should return on another night. So by the time Dogberry arrives at work, he sees no more activity.

Curious about the unusual attack, though, Dogberry begins analyzing his computer logs and is able to retrieve the source address from the hacker's FIN packets. With this information, he sends an e-mail to Abednego's ISP, advising the firm of the break-in attempt and asking for details about Abednego's account. But the system administrator at the ISP rejects Dogberry's request, citing a confidentiality policy, because merely running a scanner breaks no law.

Three evenings later Abednego resumes the hunt. But when his computer dials into his account, he finds out his



CORE DUMP can be used by hackers to obtain secret information. When a program running on a computer fails, it sometimes causes the machine to dump, or flush, the contents of a part of its random-access memory (RAM). A hacker can force such an incident to occur so that he can then sift through the discarded data, which might contain important information, such as the passwords for specific accounts on the network system.

password is no longer good. Upset, he phones the ISP and learns that his account has been shut down because of the FIN scan. Yet this turn of events does little to discourage him. In fact, he is now even more determined.

With his credit-card number and a telephone call to a different ISP, he is back on-line within minutes. This time, though, Abednego is more cautious. Through this new account, he logs on to one of his hacked accounts at yet another ISP. Once there, he gives the simple command "whois refrigerus.com." The response tells him the domain name belongs to Refrigerators R Us, a national retail chain.

Next, Abednego tries to log on to refrigerus.com through the 31,659 port by issuing the command "telnet refrigerus.com 31,659." The response is, "You lamer! Did you really think this was a back door?!" Then the 31,659 daemon attempts to crash his PC by sending corrupt packets, while e-mailing the system administrator at Abednego's hacked ISP that someone has attempted to commit a computer crime. Within minutes, Abednego's connection dies.

More determined, Abednego now tries to tiptoe around the firewall instead of forcing his way through it. Using yet another of his many hacked accounts, he begins by attempting to catalogue the computers that belong to refrigerus.com. To obtain this information, he tries "nslookup," which initiates a search throughout the Internet for master databases containing directories of IP addresses.

But "nslookup" is unable to retrieve anything useful. Dogberry must have set up the refrigerus.com network so that all packets destined for any of its internal addresses are sent first to a name-server program, which then directs them to the appropriate computers within the network. This process hinders anyone on the outside from learning details about the computers inside the firewall.

Abednego's next attempt is through an IP address scanner. First, he converts refrigerus.com to a numerical address, using "nslookup." With that number as a starting place, he scans the IP addresses above and below it. He discovers some 50 Internet host computers. Although there is no guarantee that these belong to refrigerus.com, Abednego knows it is a good bet they do.

Next, he uses "whois" to ask whether any other domain names are registered to Refrigerators R Us. The response reveals another: refrigeratorz.com, with an address that is numerically distant from that of refrigerus.com. The IP address scanner soon reveals five additional Internet hosts on numbers nearby refrigeratorz.com.

HACKER LEXICON

Abednego—A biblical Israelite held in Babylonian captivity who walked through a wall of fire and survived.

ACK—See illustration on page 72.

Back door—A secret way to enter a computer that bypasses normal security procedures.

Buffer overflow—See illustration on page 74.

Core dump—See illustration on this page.

Daemon—An automatic utility program that runs in the background of a computer.

Dogberry—The constable in William Shakespeare's *Much Ado about Nothing*.

FIN—See illustration on page 72.

Firewall—Defensive software that protects a computer system from unauthorized intruders.

FTP—File transfer protocol, a common protocol and program used to transfer files over the Internet.

IP—Internet Protocol, a low-level convention that allows computers to move packets of data across the Internet.

Internet Relay Chat—An on-line chat service.

ISP—Internet service provider.

Keystroke logger—A program that records everything a user types at a keyboard.

Port—A connection, or channel, into a computer.

RAM—Random-access memory.

Root—The highest level of access to a Unix computer.

Root kit—A program that hackers implant in a victim computer to hide their nefarious activities.

RST—See illustration on page 72.

Scanner—A program that attempts to learn about the weaknesses of a victim computer by repeatedly probing it with requests for information.

Sequence number—A number used to coordinate an upcoming IP transmission.

Shell—A software layer that provides the interface between a user and the operating system of a computer.

Sniffer—A program that records computer and network activity.

Spoof—See illustration on page 76.

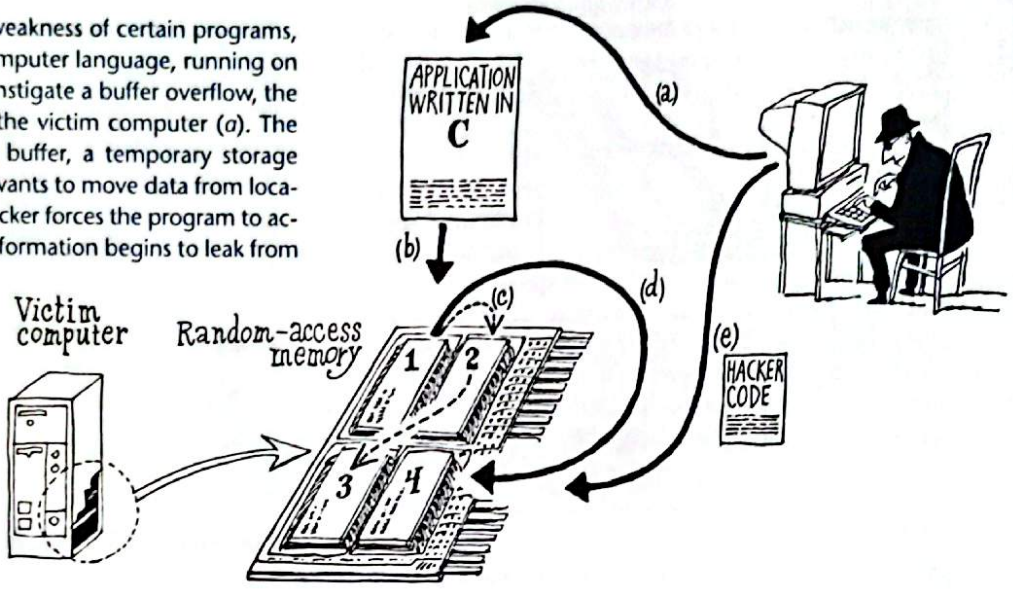
SYN—See illustration on page 72.

TCP—Transmission control protocol, the set of communications conventions that enable the sending and receiving of data over the Internet.

Telnet—A Unix command that enables a user to log on to a computer from a remote location.

Unix—A powerful operating system.

War dialer—A program that will automatically dial a range of telephone numbers.



BUFFER OVERFLOW is an exploitable weakness of certain programs, for example, those written in the C computer language, running on an operating system such as Unix. To instigate a buffer overflow, the hacker might run a C application on the victim computer (*a*). The program begins to write data into a buffer, a temporary storage space in memory (*b*). The application wants to move data from location 1 into 2, then into 3 (*c*). But the hacker forces the program to accept excess data so that some of the information begins to leak from location 1 into 4 (*d*). The hacker can take advantage of the overflow to insert his own code (*e*), which has been written to help him gain high-level privileges to the victim computer.

As a safety precaution, Abednego telnets from his current hacked account into another of his pirated accounts. He then telnets from that location to yet another account that he has hacked, remotely logging on to it in preparation to run more FIN port scans. The extra steps will force anyone in law enforcement to obtain search warrants for three companies, encumbering the process.

He also decides to hide on the third hacked computer under the protection of a root kit, a Trojan horse program that, despite its harmless appearance, will automatically delete any evidence of his actions from the logs used to detect abnormal activities. The software also defeats other programs that seek to detect alterations to system files on that computer. A root kit will even prevent people from determining that he is logged on and running programs.

From this safe perch, Abednego scans one after another of the Internet host computers at refrigerus.com and refrigeratorz.com. The FIN scanner slips straight through the firewall to every one of them. The activity, though, is detected by the EtherPeek sniffer, which again sets off Dogberry's beeper.

A haggard Dogberry, after rushing to work, soon identifies the origin of the FIN scans and alerts the system administrator at Abednego's third hacked account. But the root kit has done its job, hiding Abednego from mystified computer operators there. Abednego boldly continues, switching from the stealth scanner to Strobe in hopes of finding an IP address that the firewall does not protect.

He succeeds only in having the refrigerus.com firewall unleash a flood of meaningless data. The sudden load finally convinces the system administrator at Abednego's hacked account that there really must be an attacker at work. She takes the drastic step of cutting the entire system off from the Internet. As his connection fizzles, Abednego realizes there is no elegant way around the firewall.

Finding a Workaholic

For each of the several dozen Internet hosts at Refrigerators R Us, Abednego guesses that there are probably many other desktop computers sitting quietly in employees' cubicles and offices. What are the chances, he muses a few nights later, that somewhere among those hundreds of users are workaholics who circumvent the company firewall by phoning into their computers from their homes to perform late-night tasks? It's simple, really, for someone to buy a modem, connect it to a computer at work and plug a phone line into it before leaving for the day.

Knowing that almost every large corporation has at least one unauthorized modem on its network, Abednego sets up ShokDial, a war-dialer program that will call each of the extensions to the phone system at Refrigerators R Us as well as other numbers within that exchange. At the headquarters building of the company, the night watchman hears the ringing of one office phone after another but thinks nothing of it.

Then, at 2:57 A.M., the war dialer pulls up a modem, and Abednego is greeted with the log-on screen of a Silicon Graphics computer: "Refrigerators R Us Marketing Department. Irix 6.3." Great, Abednego thinks, because Irix is a type of Unix, which means he has found a potent portal into Dogberry's world.

Abednego's next strategy is to try brute force, using a program that will repeatedly dial the Irix box and guess passwords for root, a top-level account (usually reserved for system administrators) from which he can run any command and access all information on that particular computer. He is hoping that the owner of the Irix machine, like many harried workaholics, has negligently allowed remote access to a root account.

The password guesser starts with common words and names and from there tries less obvious choices. The slow, painstaking process can take months, even years, as the program exhausts every word in an unabridged dictionary, all names in an encyclopedia and each entry from a local phone book. But Abednego gets lucky. Around 5 A.M. he learns that the password is simply "nancy."

"Yes!" Abednego shouts as he logs on to a root shell, from which he can then issue other commands to run on that machine. Next, he secures his beachhead, using FTP (file transfer protocol) to plant a root kit and sniffer onto his latest victim. He sets the program to capture and record everything typed in at the console (a process known as keystroke logging), as well as any log-on sessions from the network. The sniffer will hide this information in an innocuously named file right there on the unwitting host. Within min-

utes, Abednego's root kit has even set up an additional way to log on: user name "revenge," password "DiEd0gB."

Abednego's last deed that morning is simple. To find the Internet address of the hijacked box, he types the "who" command, and his computer shows user "revenge" logged on to picasso.refrigeratorz.com. Later that morning the rightful owner of picasso logs on and sees no indication that someone has usurped control of her computer. Abednego's root kit is doing its job.

For Dogberry's part, all that his log reveals is an early-morning attempt to enter refrigeratorz.com from the Internet. Remembering the recent FIN scans, Dogberry is troubled by this latest incident, but he has too little information to take action.

Two nights later Abednego dials in and connects with picasso to view his logs. To his dismay, he sees that information on the internal network traffic has been encrypted. But the keystroke logger of his sniffer has recorded that someone on picasso had logged on to another computer named fantasia. Abednego now owns a user name and password for fantasia. Open sesame!

Abednego discovers that the computer is a SPARC workstation used for rendering animated sequences, perhaps for television ads. Because the box is probably a server used by many other computers, Abednego begins hunting for a password file, hoping that some of the passwords he finds will also work on other machines inside the company network.

He locates the file but discovers only "x" characters where the encrypted passwords should have been. Apparently, the information he seeks is hidden elsewhere in a shadowed file. Smiling to himself, Abednego runs the FTP program and tricks it into crashing. Bingo, core dump!

Fantasia is forced to flush a part of its random-access memory (RAM). Fortunately for Abednego, the discarded information—a record of what was being held in that RAM sector at that moment—ends up in the user directory.

The legitimate purpose of a core dump is to enable programmers to perform an autopsy on the digital remains in search of clues to a program's failure. But, as Abednego well knows, a core dump has other uses. A shadowed password system sometimes places encrypted passwords in RAM. When a person logs on, the computer does a one-way encryption of the password the user attempts and compares that with the encrypted password from the shadowed file. If the two match, the person gets in.

The shadowed password file that Abednego is able to retrieve from the core dump on fantasia is encrypted, so he starts running his password cracker. The program could be busy for the next few days, maybe even weeks.

Too impatient to wait, Abednego is already working on his next maneuver—exploiting a common vulnerability of Unix. When a program running on that operating system pours excessive data into a buffer (a temporary storage space in memory), the information will leak, infiltrating other areas of the computer's memory.

Abednego takes advantage of the buffer overflow by using it to insinuate his own code into the SPARC. The added software helps him create a root shell, from which he can then run other commands and programs. Pleased with his latest effort, Abednego next installs a root kit and sniffer. Because the kit will hide evidence of his activities only from the time when the program was activated, Abednego must mop up by deleting previous actions of his busy night.

One task remains. Is there anyone who is allowed to log on to fantasia from the Internet? Abednego types the "last" command to display records of connections people may have made to fantasia. He perks up as he sees that user names vangogh and nancy have recently entered fantasia from the Internet through the domain "adagency.com," which lies outside the Refrigerators R Us firewall.

Abednego can hardly fall asleep that morning. His adrenaline flowing, he buzzes with the knowledge that he will soon "own" Refrigerators R Us.

Closing in for the Kill

The next evening Abednego makes short work of breaking into adagency.com. At first he uses IP spoofing to trick that computer into recording a false IP address for his location. By probing adagency.com with SYN packets to elicit ACK/SYN responses with an assortment of sequence numbers, Abednego's program is able to tease out a pattern from which he can then guess the next sequence numbers and use that knowledge to fake his origin. Abednego quickly installs a sniffer on adagency.com and uses a secure-shell program to create an encrypted connection for logging on to fantasia.

From that computer, he types the "netstat" command to view tables of active connections within the network. He discovers a computer that he had missed in his earlier search. Its name, "admin.refrigerus.com," is promising. Could that be from where Dogberry oversees the system?

Meanwhile every time Abednego's PC cracks yet another combination of user name and password, he tries it on various refrigerus.com computers. But none of them works anywhere except on fantasia, which he already "owns."

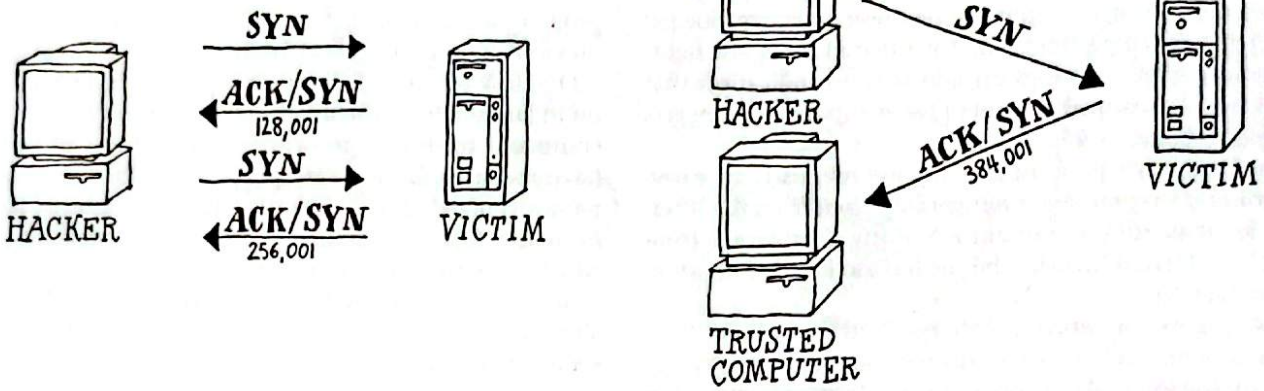
Then Abednego hits the jackpot. Twice.

On fantasia he captures keystrokes made by vangogh as that user updated the company's Web server. Now Abednego has the password he needs to hack the Refrigerators R Us Web site. In addition, his sniffer on picasso reveals that someone, Nancy, has dialed into that computer and from there used a back door to log on to a root account, hidden by her root kit, at admin.refrigerus.com.

He slips right behind Nancy into admin.refrigerus.com. Using the root account there, he tries logging on to one Refrigerators R Us computer after another. Dogberry, however, has been exceedingly careful. On the Refrigerators R Us network, even root privileges do not allow someone to enter other computers without providing new passwords.

Only briefly distracted, he turns his attention back to the Web server and logs on to it using his recently acquired password. From his home PC, he then uploads a new version of the Refrigerators R Us home page that he had put together in anticipation of this day.

Back at Refrigerators R Us, Dogberry is working late, poring over his logs. It seems the marketing people have been



IP SPOOFING enables a hacker to fake his identity. The hacker first probes his victim by sending multiple SYN packets [*see illustration on page 72*] to obtain ACK/SYN messages with sequence numbers (*left*). From these responses, the hacker is able to uncover a pattern. In this example, he notices that the numbers increase by an increment of 128,000. Next, the hacker sends a SYN that impersonates another computer that the victim trusts. The victim then transmits an ACK/SYN to this authorized host (*center*). Although the hacker does not receive this particular response, he can nonetheless continue the correspondence as if he had: he issues

getting an unusual number of connections from adagency.com. Tomorrow he will ask those folks exactly what is going on. He will also call the system administrator at adagency.com, a colleague whom he once helped to install some new system software.

Just as Dogberry is about to head home for the night, the phone in his office rings. An angry customer complains that Refrigerators R Us's Web site features a pornographic movie with a refrigerator as a prop. After bringing up and viewing the defaced Web page, Dogberry moves quickly to sever the umbilical Ethernet cable that connects the company network to the Internet.

Abednego is enraged when his obscene masterpiece is taken down so quickly. But he is also worried that he has left too much evidence behind, so he returns using the dial-up line to picasso—an entryway that is still unknown to Dogberry. He buys time by reformatting completely the administrative computer's hard disk, which shuts down the company network, temporarily thwarting Dogberry's efforts to gather details of the attack.

Dogberry rushes to the administrative computer with hopes to reboot it from the console, but he is too late. Dogberry must now rebuild the software on that computer from scratch. (Unbeknownst to Abednego, though, the EtherPeek sniffer running on a nearby Macintosh has also been making logs.)

Abednego, still peeved about the Web site, has one final act that night: he unleashes a flood of data packets against refrigerus.com. Soon Dogberry gets a frantic call from a company salesperson who, using her laptop PC and a phone line in her hotel room, wants to retrieve her important e-mail but has been unable to connect to the mail server at Refrigerators R Us.

The next morning an exhausted Dogberry begs the vice president of technology at Refrigerators R Us for an okay to wipe clean every computer in the network, reinstall every program and change all passwords. But the extensive—though prudent—measure would require shutting the system down for days, and the vice president denies the request.

At this point, Abednego's malicious and destructive exploits have gone well past the legal bounds for hacking. But the FBI, which is severely understaffed, has been busy investigating some recent break-ins at several army and navy computer systems around the U.S. Dogberry will have to gather more evidence himself.

Because the attacker remained on the system even after it had been physically disconnected from the Internet, Dogberry suspects there must be a contraband modem somewhere in the building. He runs his own war dialer and discovers the culprit. He will soon have words with the marketing department!

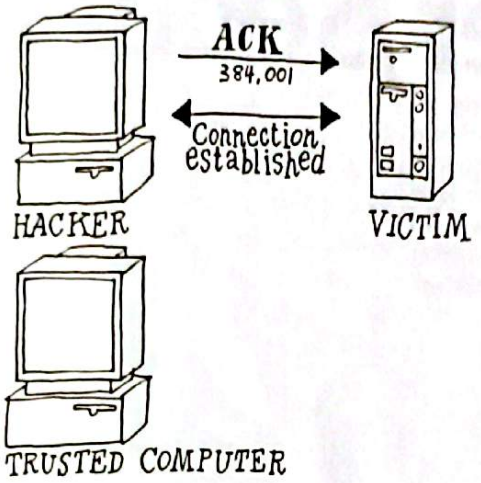
Dogberry then reloads a clean version of his main administrative computer. Next, on a Windows NT server that Dogberry knows has not been tampered with, he deploys T-sight, an advanced antihacker program that can monitor every machine on the company network.

Last, Dogberry sets his trap. T-sight will watch for the attacker's next connection to admin.refrigerus.com and will redirect the intruder into a "jail" computer. Once there, the culprit can be monitored and traced. To keep the unsuspecting person distracted, Dogberry enlists a team of programmers to make the jail look like an accounting system, complete with the tempting bait of fake financial data.

Pride Goeth Before...

Just two nights later Dogberry is standing watch at 8:17 P.M. when he discovers someone once again entering admin.refrigerus.com. It is Abednego. Why has he returned so soon? Abednego was exhilarated when he learned that his pornographic Web site had become the talk of the hacker underground. He had even rated a brief mention on CNN. The publicity and his hubris were a potent combination, making Abednego feel invincible.

In fact, tonight he has brazenly reentered Refrigerators R Us without his customary caution. After dialing into a guest account on an ISP, he telnetted directly to adagency.com to gain faster access to fantasia's back door.



an ACK message with the correct predicted sequence number, thus establishing a connection between his computer and the victim (*right*). The hacker can then transmit information that the victim will assume is benign because of the mistaken belief that it is coming from the trusted host computer.

From admin.refrigerus.com, Abednego is lured to the jail by T-sight. He can hardly control his excitement as he begins sifting through what he believes are sensitive financial records.

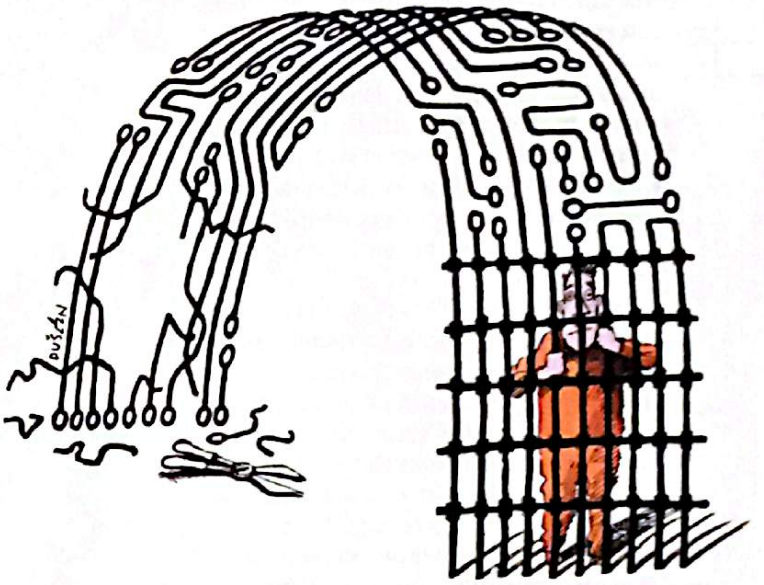
Dogberry, too, is busy. Quickly analyzing data from T-sight, he obtains Abednego's root password on fantasia—DiEd0gB—and is able to trace the intruder back to adagency.com. Dogberry calls the pager of the system administrator there. She has already left work, but she phones Dogberry from a restaurant to help him continue tracking Abednego.

So while Abednego is retrieving a huge file containing bogus credit-card numbers, Dogberry installs a sniffer on adagency.com. He is even able to sneak unnoticed into Abednego's account on that computer by typing DiEd0gB, because Abednego has lazily used the same password for all his root kits. Then, just minutes before Abednego finishes his download and logs off, Dogberry succeeds in tracking the trail of the plundered credit-card file back to Abednego's dial-up account at the ISP.

The information Dogberry has obtained is enough to bring in the FBI, which contacts the ISP the next day to obtain Abednego's identity from the company's phone logs. With enough evidence in hand, including the Macintosh's high-quality EtherPeek logs, the U.S. Attorney's office approves a search warrant.

Soon after, FBI agents raid Abednego's apartment and confiscate his PC. The hard disk of the computer will reveal all. Abednego had taken the precaution of erasing incriminating files from his PC after each night's escapade. He is chagrined to learn that the FBI can extract that information from his hard drive even after it had been erased and overwritten several times. Soon a laboratory has recovered details of his past trespasses, including the time he romped through the computer system at a major banking institution in the Northeast.

The megabytes of incriminating data provide the smoking gun necessary to indict Abednego on multiple counts of computer fraud. Unfortunately for him, the trial judge assigned to his case is known for her tough stance on cybercrime. Taking his attorney's advice, Abednego wisely accepts a plea bargain even though, like many hackers who have crossed the line, he insists that his activities—which, for Refrigerators R Us alone, resulted in thousands of dollars in damages—were just playful pranks. Abednego is currently serving a two-year sentence in a federal prison.



The Author

CAROLYN P. MEINEL is addicted to the frontiers of technology. She is the author of *The Happy Hacker: A Guide to (Mostly) Harmless Computer Hacking* (American Eagle Publications, 1998) and the president of Happy Hacker, Inc., a nonprofit organization devoted to teaching people how to hack responsibly and legally. The group runs a hacker wargame on its World Wide Web site (http://www.happyhacker.org). Meinel holds an M.S. in industrial engineering from the University of Arizona. She is currently working on a book with Jason Chapman, recounting her many adventures battling hackers. Meinel wishes to acknowledge the help of Michael and Diana Neuman, Damian Bates, Emilio Gomez, Mahboud Zabetian and Mark Schmitz in researching this manuscript.

Further Reading

ESSENTIAL SYSTEM ADMINISTRATION. Second edition. Æleen Frisch. O'Reilly & Associates, Sebastopol, Calif., 1995.

INTERNET FIREWALLS AND NETWORK SECURITY. Second edition. Chris Hare and Karanjit Siyan. New Riders Publishing, Indianapolis, 1996.

MAXIMUM SECURITY: A HACKER'S GUIDE TO PROTECTING YOUR INTERNET SITE AND NETWORK. Anonymous. Sams Publishing, Indianapolis, 1997.

THE GIANT BLACK BOOK OF COMPUTER VIRUSES. Second edition. Mark Ludwig. American Eagle Publications, Show Low, Ariz., 1998.

Additional information can be obtained at http://www.geek-girl.com/bugtraq, http://ntbugtraq.ntadvice.com, http://rootshell.com, http://www.infowar.com, http://www.antionline.com and http://www.happyhacker.org on the World Wide Web.

Details of EtherPeek and T-sight can be obtained at http://www.aggroup.com and http://www.engarde.com, respectively.

How Computer Security

Three types of safeguards offer a formidable defense against Internet intruders

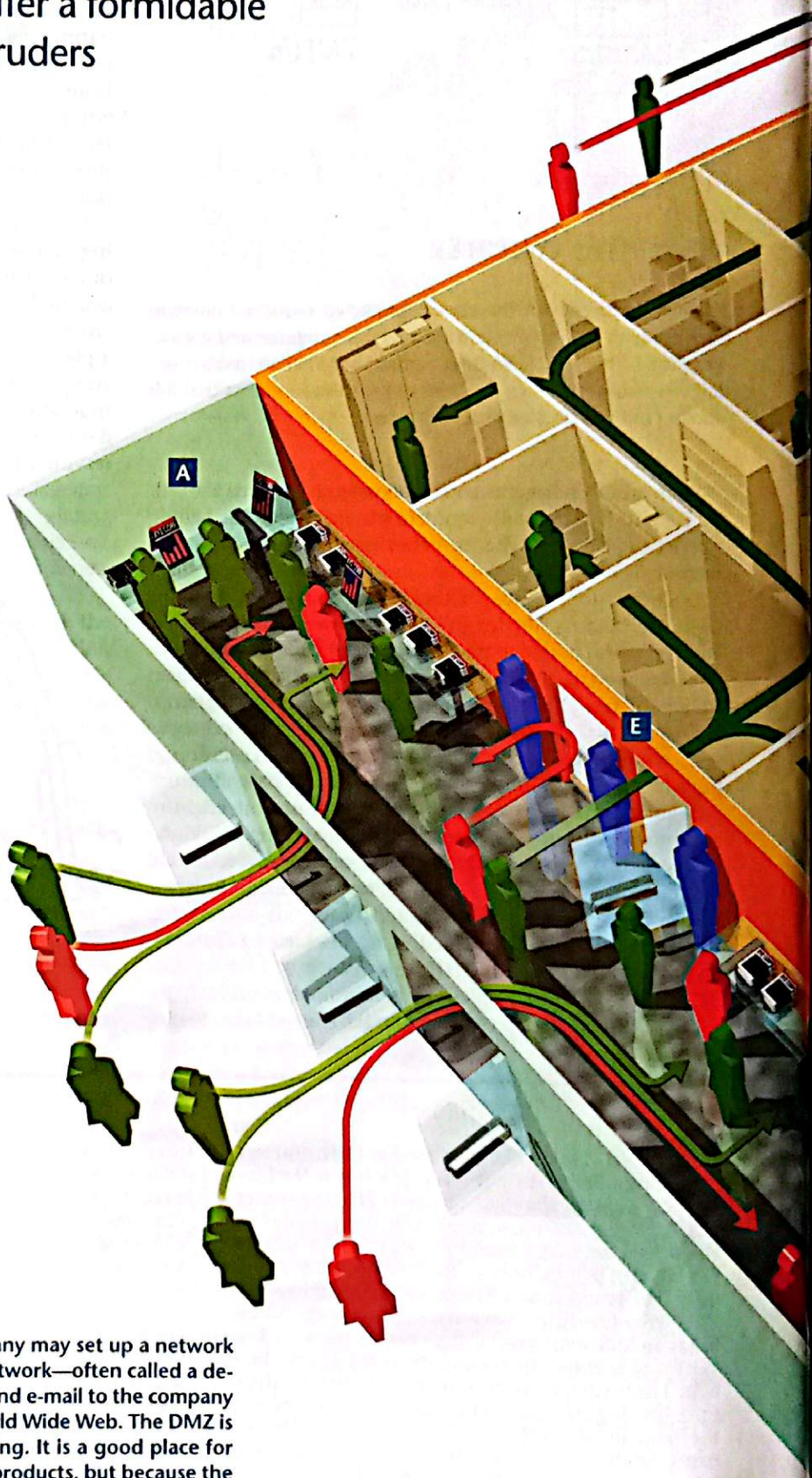
1 Firewalls

by William Cheswick, *Bell Labs (division of Lucent Technologies)* and Steven M. Bellovin, *AT&T Research*

Computer networks will always be vulnerable to attack. As long as companies use the Internet—for transferring files, sending e-mail, downloading programs and so on—there will always be the chance that some malicious outsider will find a way to wreak havoc with their computer systems. But there are ways to make a network much more resistant to attack. The first line of defense is the firewall, a software program that acts as a gatekeeper between the Internet and a company's "intranet"—the network of computers used by the company's employees.

The two most common kinds of firewalls are packet filters and application-level firewalls. A packet filter, which typically runs on a machine called a router, examines the source address and destination address of every packet of data going in or out of the company's network. The filter can block packets from certain addresses from entering the network—and prevent other packets from leaving. An application-level firewall examines the content of the Internet traffic as well as the addresses; it is slower than a packet filter, but it allows the company to implement a more detailed security policy.

In the illustration, the maze of offices represents a computer network protected by firewalls. The green figures symbolize authorized packets of data. The red figures are potentially harmful packets that should not be allowed into the network.



A

Outside the firewall (*outlined in orange*), a company may set up a network that is accessible to the general public. Such a network—often called a demilitarized zone, or DMZ—allows customers to send e-mail to the company or browse through the company's site on the World Wide Web. The DMZ is analogous to the public lobby of an office building. It is a good place for customers to get information on the company's products, but because the DMZ is open to anyone, sensitive data should not be placed there.

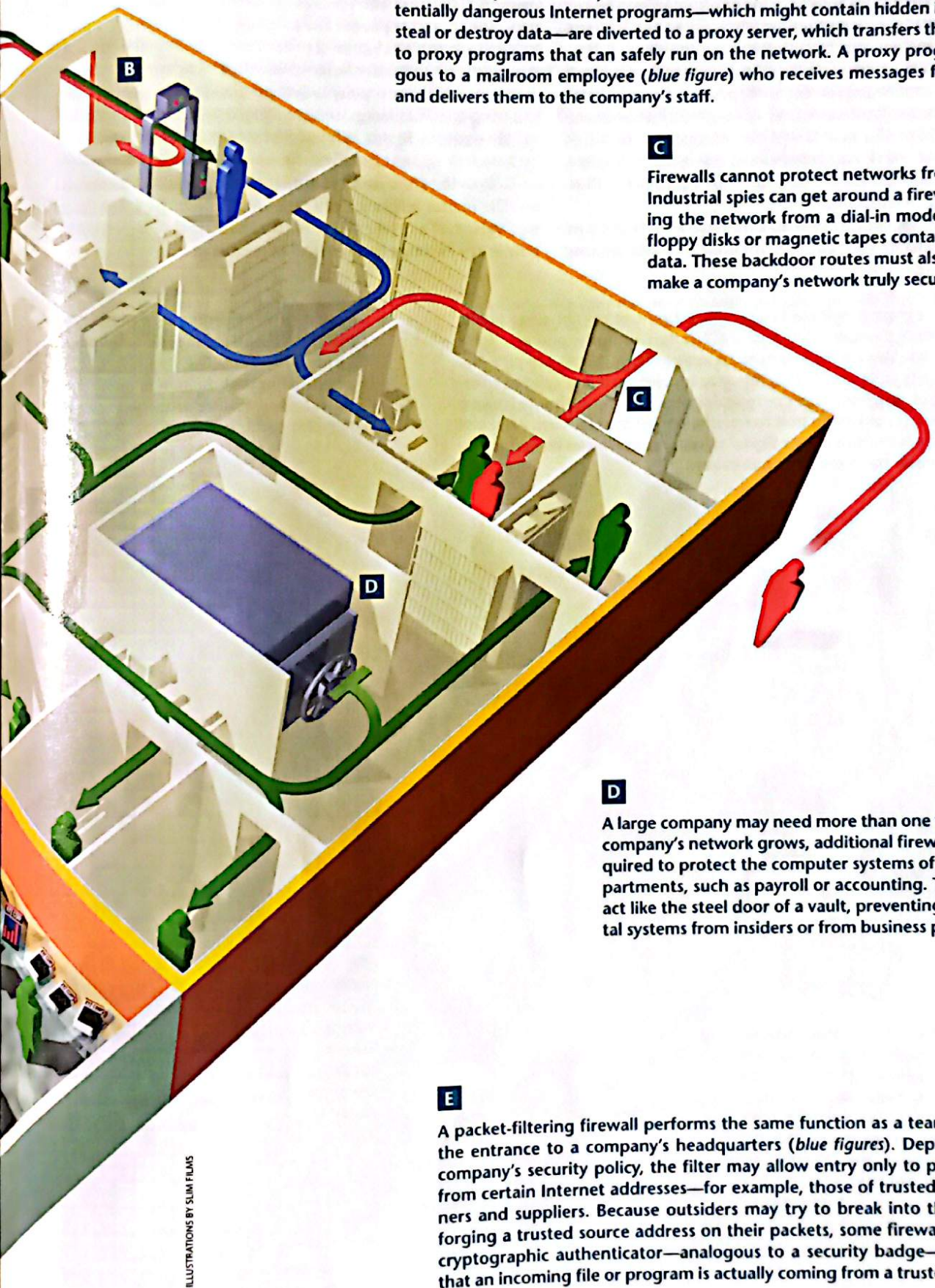
Works

B

An application-level firewall is like the company's mailroom: it can scan incoming e-mail for computer viruses just as mailroom employees can x-ray bulky packages. Potentially dangerous Internet programs—which might contain hidden instructions to steal or destroy data—are diverted to a proxy server, which transfers the information to proxy programs that can safely run on the network. A proxy program is analogous to a mailroom employee (*blue figure*) who receives messages from outsiders and delivers them to the company's staff.

C

Firewalls cannot protect networks from all attacks. Industrial spies can get around a firewall by accessing the network from a dial-in modem or stealing floppy disks or magnetic tapes containing sensitive data. These backdoor routes must also be closed to make a company's network truly secure.



D

A large company may need more than one firewall. As the company's network grows, additional firewalls may be required to protect the computer systems of important departments, such as payroll or accounting. These firewalls act like the steel door of a vault, preventing attacks on vital systems from insiders or from business partners.

E

A packet-filtering firewall performs the same function as a team of guards at the entrance to a company's headquarters (*blue figures*). Depending on the company's security policy, the filter may allow entry only to packets coming from certain Internet addresses—for example, those of trusted business partners and suppliers. Because outsiders may try to break into the network by forging a trusted source address on their packets, some firewalls search for a cryptographic authenticator—analogous to a security badge—which verifies that an incoming file or program is actually coming from a trusted address.

ALL ILLUSTRATIONS BY SLIM FILMS

2 Digital Certificates

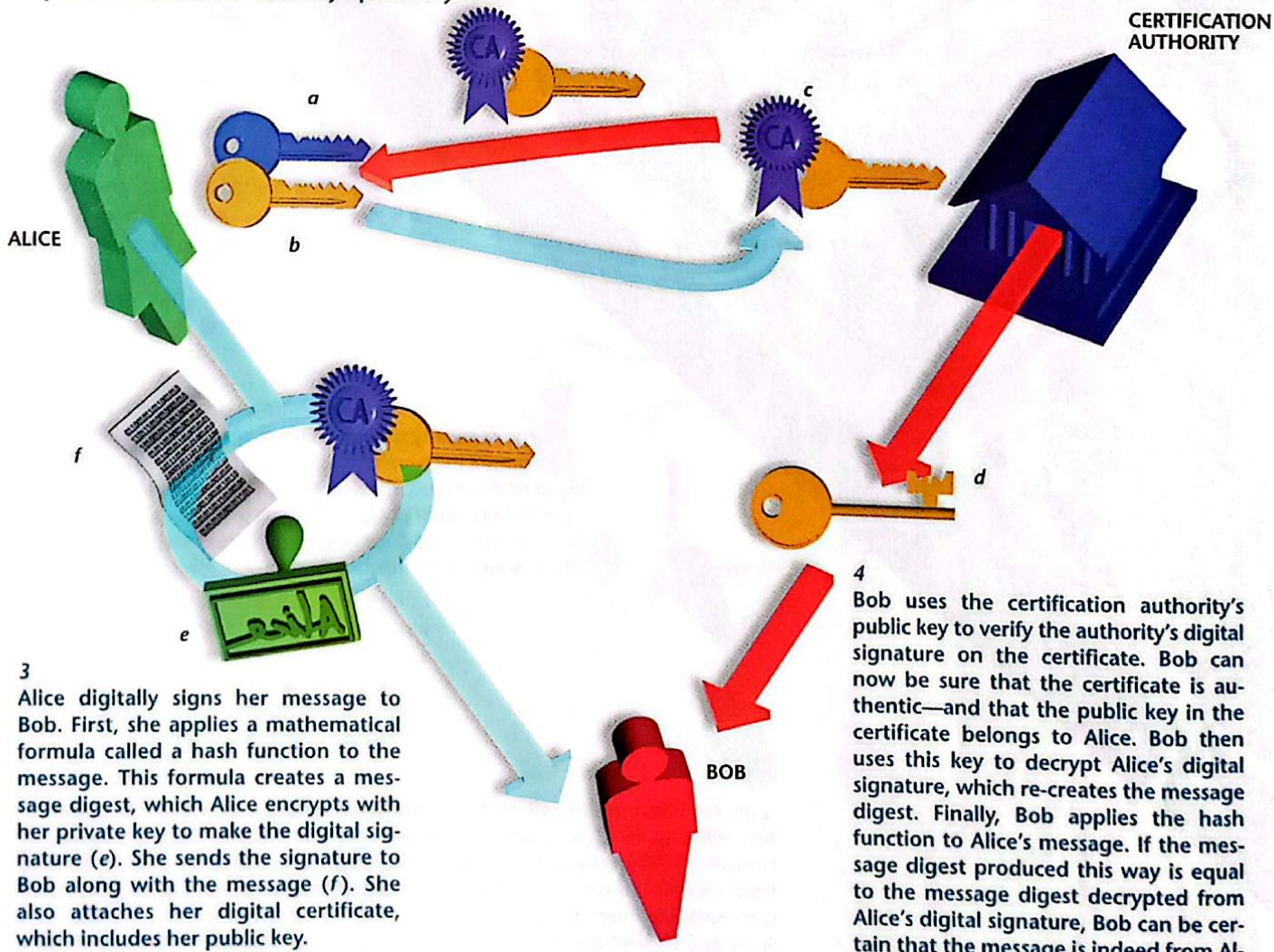
by Warwick Ford, *VeriSign*

Digital certificates play an essential role in public-key cryptography, a method widely used on the Internet to keep communications secure. To send and receive messages with this method, a computer user must have a pair of cryptographic keys—a private key and a public key—which are long strings of data, usually containing 500 to 1,000 bits. The user keeps the private key in a safe place—encrypted on a computer's hard drive, for example—but makes the public key known to the people with whom he or she wants to communicate.

Let's say that Alice wants to send a message to Bob. Because she wants Bob to be sure that the message is really coming from her, Alice uses her private key to create a digital signature, which accompanies the message. Bob uses Alice's public key to verify the signature. But how can Bob be sure that the public key actually belongs to Alice? An impostor could create her own key pair and send the public key to Bob, claiming that it belongs to Alice. To prevent such a possibility, Alice must obtain a digital certificate, a data item issued by a widely trusted certification authority, such as VeriSign or GTE CyberTrust or an authority set up by Alice's company. The digital certificate can be thought of as the cyberspace equivalent of a driver's license. It confirms that a particular public key belongs to a particular person or entity.

1
Alice uses cryptographic software to generate a private key (*a*) and a public key (*b*). She sends the public key to a certification authority and asks for a digital certificate. The authority needs to authenticate Alice's identity; depending on the type of certificate, this may involve verifying private information that Alice supplies. If her credentials check out, the authority issues a digital certificate (*c*) affirming that the public key belongs to Alice. Attached to the certificate is the authority's digital signature, which can be verified by anyone who knows the authority's public key.

2
The certification authority's public key (*d*) is distributed to anyone who needs it, including Bob. The key is typically embedded in Web browsers and other application software used for secure computer communications.



3
Alice digitally signs her message to Bob. First, she applies a mathematical formula called a hash function to the message. This formula creates a message digest, which Alice encrypts with her private key to make the digital signature (*e*). She sends the signature to Bob along with the message (*f*). She also attaches her digital certificate, which includes her public key.

4
Bob uses the certification authority's public key to verify the authority's digital signature on the certificate. Bob can now be sure that the certificate is authentic—and that the public key in the certificate belongs to Alice. Bob then uses this key to decrypt Alice's digital signature, which re-creates the message digest. Finally, Bob applies the hash function to Alice's message. If the message digest produced this way is equal to the message digest decrypted from Alice's digital signature, Bob can be certain that the message is indeed from Alice and that it has not been tampered with by anyone else on the network.

3 The Java Sandbox

by James Gosling, *Sun Microsystems*

The Java programming language can be used by software developers to write small applications—called applets—that can be downloaded from the Internet or other computer networks. The danger is that an unscrupulous person might create an applet that would tamper with a user's computer system by erasing files, stealing data or passing along a computer virus. But the Java language has been designed to prevent such breaches.

The key to Java's security is a layer of software called the Java Virtual Machine, which is needed to execute any applet written in the programming language. When a computer user downloads an applet, the virtual machine initially prevents the program from gaining access to the computer's hard drive, network connections and other vital system resources. At this stage the applet can be imagined as sitting in a child's sandbox, a place where it can do no damage. An applet can get out of the sandbox only if the virtual machine verifies that the program comes from a trusted source.

1
A computer user named Sam visits a bank's site on the World Wide Web and finds an applet that would help him calculate his mortgage payments. When Sam downloads the applet from the bank's Web server, the Java Virtual Machine—which is embedded in Sam's Web browser—allows the applet into the RAM chips of Sam's computer but blocks the applet's access to the computer's hard drive. The applet is in the sandbox.

2
First, a byte-code verifier determines whether the applet is written in legitimate Java code. If the applet is not written correctly, Sam will not be allowed to use the program at all. Then the virtual machine looks for a digital signature attached to the applet. The signature identifies the person or entity who created the applet and reveals whether a third party has altered the program. The applet must remain in the sandbox if it does not have a verifiable signature. Sam can use the program to perform calculations but not to read or write files on his hard drive.

3
If the applet's signature is verified, the virtual machine determines how much access to give the program. Sam can adjust the amount of access according to his security needs. For example, he can allow the applet to read any file on his hard drive or bar it from parts of the drive that contain confidential data.

Cryptography for the Internet

E-mail and other information sent electronically are like digital postcards—they afford little privacy. Well-designed cryptography systems can ensure the secrecy of such transmissions

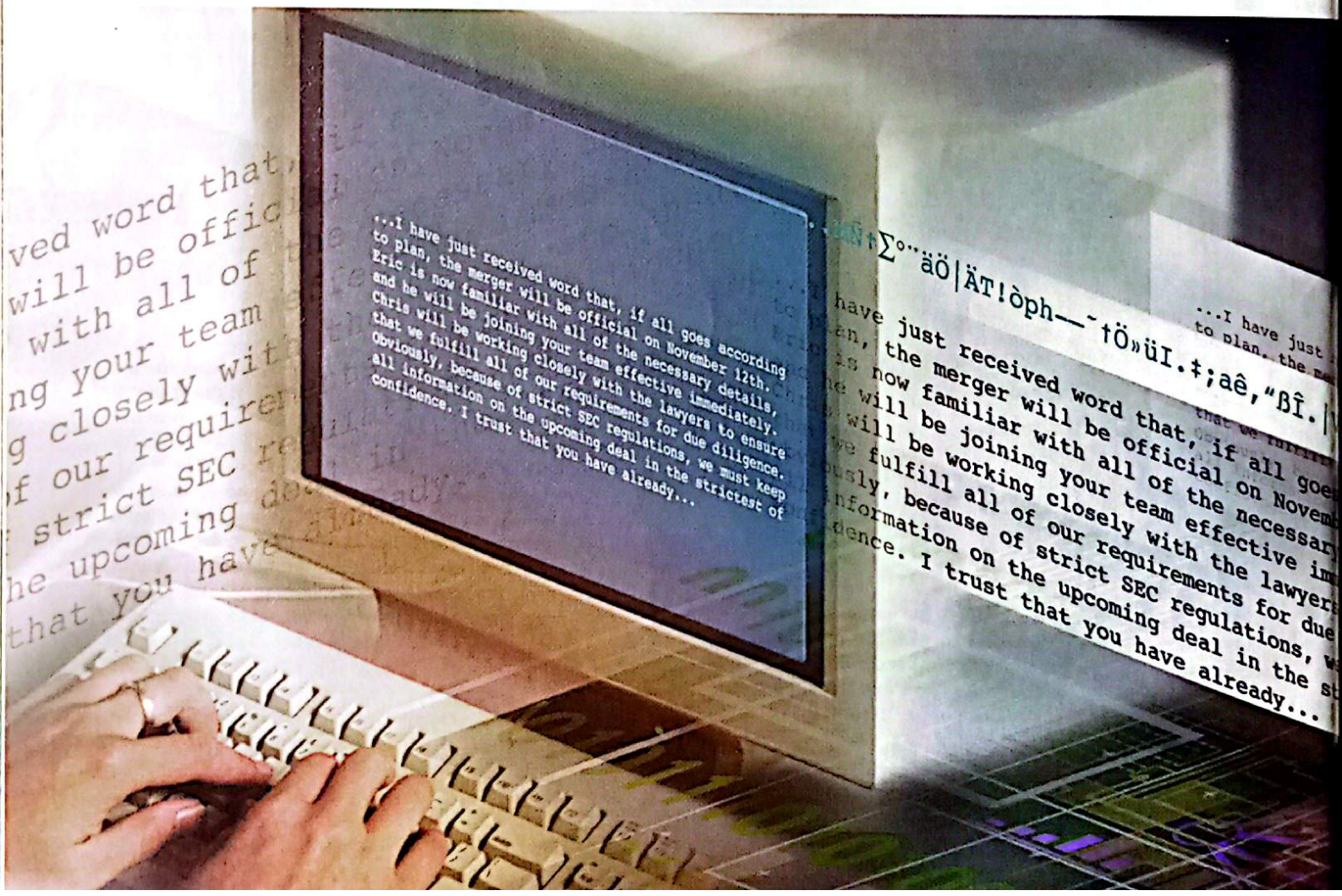
by Philip R. Zimmermann

Sending letters through the post office might take days, but at least the correspondence is guaranteed some degree of privacy. E-mail delivered over the Internet, on the other hand, can be blindingly fast but is highly susceptible to electronic eavesdroppers. One way to increase the privacy of such transmissions is to encrypt them, scrambling the information in complex ways to render it unintelligible to anyone but the intended recipient.

Since the 1980s the development of sophisticated algorithms and fast but affordable computer hardware have made powerful, military-grade cryptographic systems available to millions of people with ordinary personal computers. Recent technological improvements promise to make such systems increasingly resistant to even the most advanced cipher-cracking techniques.

Out from the Shadows

Four decades ago the Pentagon's requirements for tiny custom circuits to fit into missiles and spacecraft were the driving force behind the U.S. electronics industry. Today civilian demands dominate, and the military currently



satisfies most of its needs with off-the-shelf products designed for the much larger consumer market. The same thing is happening with cryptography.

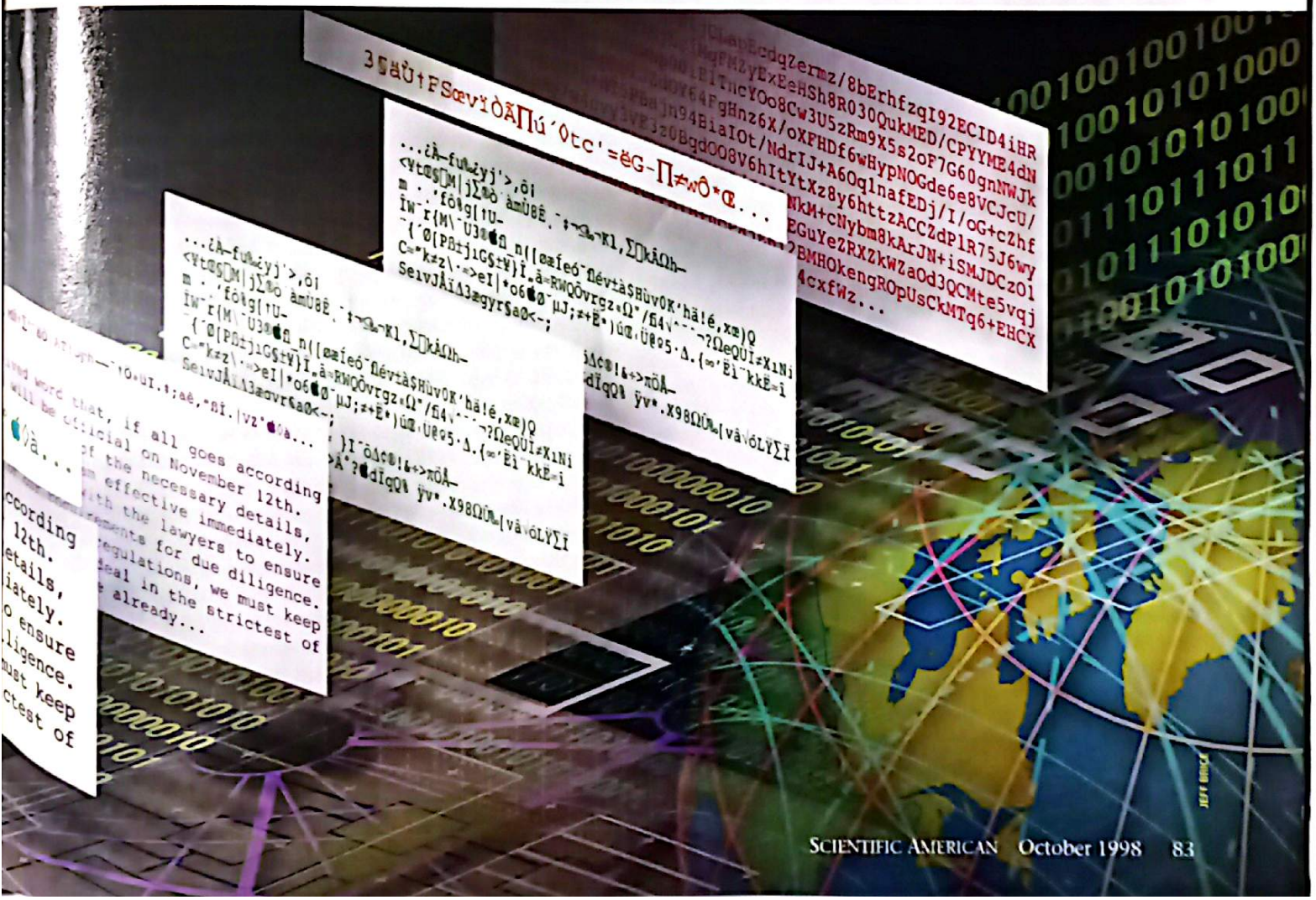
Until the mid-1970s the National Security Agency (NSA) had a virtual monopoly on U.S. encryption technology, a field that was kept shrouded in secrecy. Then, in 1976, the seminal article "New Directions in Cryptography," in which Whitfield Diffie and Martin E. Hellman of Stanford University first described "public-key cryptography" in the open literature, forever changed the landscape. In the years since that publication, an energetic cryptographic community in academia and industry has emerged, publishing an ever increasing number of papers and building a mature discipline. The growing popularity of the Internet—and people's concerns about the privacy of that medium—has only intensified the trend. Today some of the best ciphers and systems are being developed by cryptographers at universities and in the private sector all over the world. In fact, the NSA is now beginning to buy commercial products for a portion of its cryptographic needs.

Why was Diffie and Hellman's introduction of public-key cryptography so crucial? In conventional cryptosystems, a single key is used for both encryption and decryption. Such systems, called symmetric, require the key to be transmitted over a secure channel—a process that is often inconvenient. After all, if a secure channel exists, why is encryption needed in the first place? This limitation hobbled cryptography.

Diffie and Hellman removed that constraint. Public-key cryptography allows the participants to communicate without requiring a secret means of delivering the keys. Such symmetric systems rely on a pair of keys that are different but complementary. Each key unlocks the message that the other key encrypts, but the process is not reversible: the key used to encrypt a message cannot be used to decrypt it. Thus, one of the complementary keys (public) can be disseminated widely, whereas the other key (private) is held only by its owner. When Bob wants to send a message to Alice, he can use her public key to encrypt the information, which she will then use her private key to decrypt.

Public-key cryptosystems are based on mathematical problems that are easy to compute in one direction but painfully slow to solve in the reverse. The two main public-key algorithms are the Diffie-Hellman (and its variants, such as the Digital Signature Standard from the National Institute of Standards and Technology, ElGamal and elliptic curve approaches) and RSA, developed at the Massachusetts Institute of Technology by computer scientists Ronald L. Rivest, Adi Shamir and Leonard M. Adleman.

ENCRYPTING A PRIVATE MESSAGE that Bob will send to Alice over the Internet requires several steps. In this conceptual schematic, Bob first computes a hash of the text [*see diagram on page 85*]. He then encrypts the hash using his private key [*see box on next page*]. The resulting information (*blue, below*) serves as Bob's "signature." Bob compresses the signature and his message electronically (*purple*) and enciphers the file (*green*) using a particular session key. Bob encrypts this key using Alice's public key, and the result (*orange*) is added to the message. Finally, the file is converted into alphanumeric characters (*red*) for transmission over the Internet. At the receiving end, the steps are essentially reversed, with Alice using her private key to decrypt the session key, which she can then use to decipher the rest of the message.



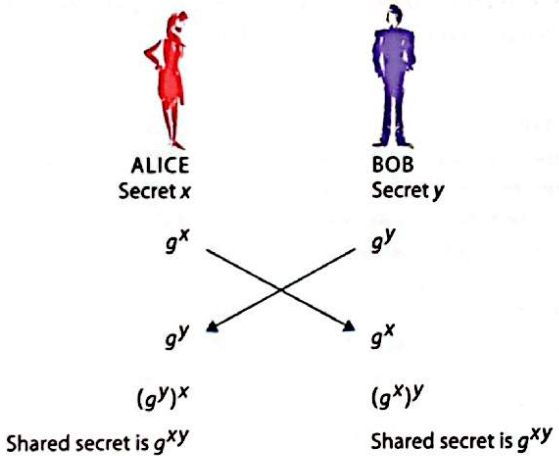
The former approach uses discrete logarithms. It is simple to compute g^x modulo p: just raise g to the x power, divide that quantity by a large prime number p, and then take the remainder of that operation. But given g, p and the value of g^x modulo p, it is infeasible to recover x [see "The Mathematics of Public-Key Cryptography," by Martin E. Hellman; SCIENTIFIC AMERICAN, August 1979].

The RSA system is based on the difficulty of factoring. It is straightforward to multiply two large prime numbers together, but it is extremely difficult to factor that huge composite back into its two primes [see "Mathematical Games," by Martin Gardner; SCIENTIFIC AMERICAN, August 1977].

Another beauty of public-key cryptography is that it can be used for message authentication: a recipient can verify the identity of the sender. When Bob transmits a message to Alice, he first encrypts it with his private key, then reencrypts the encrypted message with Alice's public key. Alice, after receiving the transmission, reverses the steps. She first decrypts the message with her own private key, then decrypts it again with Bob's public key. If the final text is legible, Alice can be confident that Bob actually wrote the message.

Of course, all this encrypting and decrypting requires myriad mathematical calculations. But software applications, such as PGP, running on PCs can automate the process. Using one of those packages, Bob and Alice need only press the "encrypt" and "decrypt" buttons on their computers, and the number crunching is performed behind the scenes.

For all its innovation, public-key cryptography has two severe limitations. First, because of its relatively slow speed, the technology is impractical for encrypting large messages. Second, and perhaps more important, public-key cryptography sometimes allows patterns in a message to survive the encryption process. The patterns are thus detectable in the enciphered text, making the technology vulnerable to cryptanalysis. (Cryptography is the science of making ciphers, cryptanalysis is the study of breaking them, and cryptology is both disciplines.)



Public-Key Cryptography

For centuries, cryptography was hampered by the so-called key-exchange problem. Specifically, if Bob wanted to send Alice an enciphered message, he also somehow had to transmit to her the secret encryption key that he had used. Public-key cryptosystems overcame this limitation by relying on clever mathematics.

In the Diffie-Hellman algorithm, which helped to spawn the field of public-key cryptography, Alice uses her secret number x to calculate g^x and sends that quantity to Bob. On his end, Bob uses his secret number y to compute g^y and sends that to Alice. (Note that the value of g is publicly known.) After Alice receives this information, she can then compute $(g^y)^x$, which is equal to $(g^x)^y$, the value that Bob calculates. This quantity can become their shared, secret encryption key.

But someone who has intercepted Alice's g^x and Bob's g^y would be able to derive the secret x and y. So to thwart any eavesdroppers, Alice and Bob insert the modulo function, which calls for the remainder from a division operation. (For example, 14 modulo 4 = 2 because the remainder of 14 divided by 4 is 2.) This added twist ensures secrecy—instead of sending g^x to Bob, Alice transmits the value of g^x modulo p, from which eavesdroppers would have great difficulty in recovering x, even if they know g and p.

With additional mathematics, the Diffie-Hellman algorithm has evolved into cryptosystems that generate two complementary keys, one private (for Alice, x) and the other public (consisting of g, p and the value of g^x modulo p). Ingeniously, the private key deciphers the message that was enciphered by the public key, but the key used to encrypt a message cannot be used to decrypt it. Thus, Bob can use Alice's public key (which she has disseminated to everyone) to encrypt a message to her, which she—and only she—can decrypt using her private key. —*P.R.Z.*

Symmetric Workhorses

Consequently, the bulk of encryption is usually performed by faster and more secure symmetric ciphers, with public-key cryptography limited to the small—but essential—function of exchanging the symmetric keys. Specifically, Bob encrypts his message with a quick and strong symmetric cipher. He then needs to send Alice the symmetric key that he used, so he enciphers it with her public key and attaches the result to his encrypted message. Alice will decrypt the symmetric key with her private key so that she can use that information to decrypt the rest of Bob's message.

For authentication, Bob again does not use public-key cryptography to "sign" his transmission directly. Instead he computes a hash, or fingerprint, of his message. Such mathematical procedures can be used to condense an input of any size into a digest of fixed length, typically 160 bits long. (A bit is the most basic unit of computer data. It stores one of two possible states, represented by 0 or 1.) Cryptographically strong hash functions, such as SHA-1, RIPEMD-160 and MD5, are designed so that a forger would find it computationally infeasible to devise a different message that would yield the same hash. In other words, the fingerprints generated are virtually unique: two different messages will almost certainly yield distinct digests.

After computing a hash of his message, Bob encrypts that information with his private key. He sends this "signature" with the rest of his encrypted transmission. Alice receives the encrypted hash and decrypts it with Bob's public key. She can then compare the result with the hash she computes herself after decrypting the message. A match proves both that the transmission has not been tampered with and that Bob is the sender.

For encrypting such information to be sent over the Internet, the most common method is to break the data into fixed-size blocks, each usually 64 or 128 bits long, so that the encryption can be performed a chunk at a time. So-

ILLUSTRATIONS BY DMITRY KRASNY

called block ciphers usually encrypt each chunk using multiple rounds (the exact number is dictated by the particular algorithm) of mathematical operations, with the output of one iteration fed as input to the next. Each round often involves both permutation (shuffling "xtv" to "tvx") and substitution (changing "tvx" to "cb2"). A section of the key helps to transform the data during the iterations.

Feeding identical chunks of text to a block cipher will lead to encrypted blocks that are identical to each other. To suppress any such block-aligned patterns from forming (which would make the cipher easier to crack), block algorithms typically use some kind of chaining. Blocks that have already been encrypted are looped back to help encrypt subsequent chunks. In effect, the encryption of a block of text depends on *all* the previous blocks.

Block ciphers have symmetric keys that are usually 56, 128 or 256 bits long. Well-known examples are the Data Encryption Standard (DES), triple-DES, CAST, IDEA and Skipjack. The workhorses of cryptography, block algorithms have become the focus of much recent research.

The Key Is the Key

The most sensitive operation in cryptography is the generation of keys. For a system to be as secure as possible, the keys should be numbers that are truly random, unpredictable by an attacker. Such numbers are different from the deterministic pseudorandom sequences that computers generate algorithmically for games and simulations. Truly random numbers can be derived only from the environmental "noise" of the physical world, such as the process of radioactive decay.

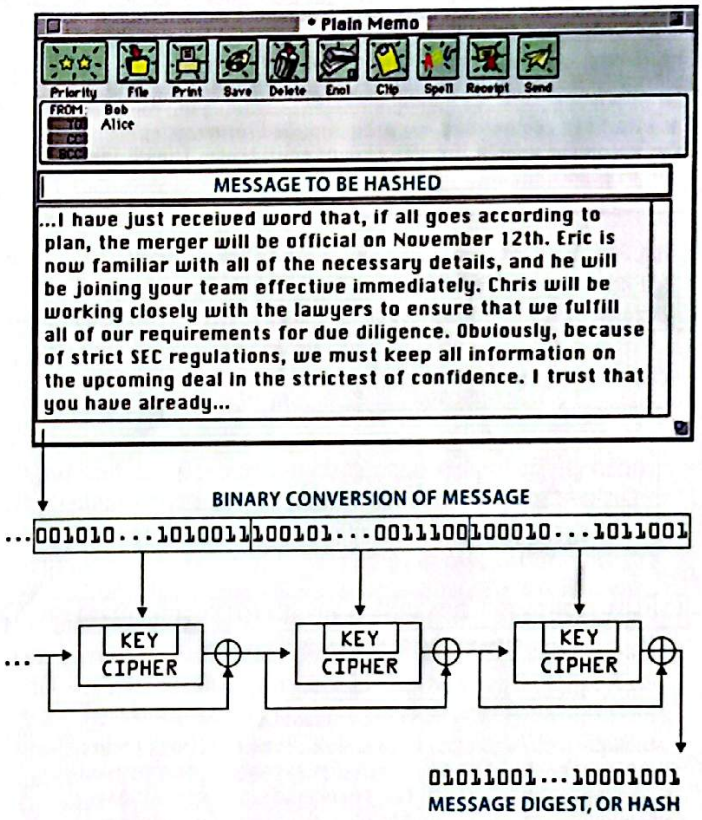
Such high-quality randomness is difficult to generate in a computer. One method is to measure the time, in microseconds, between each human-supplied keystroke, which is impossible to predict. Data gathered in this way are not quite random enough for generating keys directly, but the information can be passed through a hash function to distill the disorder.

Interestingly, the only cipher that cryptologists have ever proved to be perfectly secure is the one-time pad (OTP), in which the key is as long as the message itself. In an OTP, a random sequence is used to encipher a message bit for bit—that is, the 34th bit of the key is used to alter the 34th bit of the message. The key must be truly random. It cannot be a pseudorandom sequence produced by a deterministic algorithm; otherwise the cipher may be crackable. OTPs are rarely used because of their impracticality: the key must be as long as the message, and it must be sent to the receiver over a secure channel. Moreover, it can be used only once, or an attacker could break the messages.

Although many people think key size is the determining factor in cryptographic strength, an equally important criterion is the quality of the cipher's design. Consider a simple substitution cipher in which all *A*s are changed to *W*s, all *B*s turned into *K*s, all *C*s transformed to *Q*s, and so on. The number of different ways to rearrange the alphabet is given by 26 factorial (that is, $26 \times 25 \times 24 \times \ldots 3 \times 2 \times 1$). That quantity is roughly equivalent to 2^{88}, a "key space" of different combinations that is regarded as fairly respectable, requiring enormous computing resources to break if every possible key must be tried. Yet when I was a kid I would crack this type of cryptogram all the time with no more than a pencil and paper. I simply looked for the most common letter and assumed it was probably *E* and then found the second most common letter and assigned *T* to it, and so on. Clearly, despite its vast key space, this type of cipher is very weak.

For a well-designed cryptography system, though, the key size does relate directly to the effort required to crack it. For block ciphers, the relation is usually exponential. Adding just one bit to the key length doubles the work the attacker must do to try all the keys. And doubling the key size squares the amount of effort. On average, a 128-bit key requires about 2^{127} (in decimal, 1.7×10^{38}) operations to break.

Public-key algorithms are less sensitive. Typically, they have subexponential but superpolynomial key spaces, which means that doubling the length of the key increases the work substantially, but the amount is less than a squaring of the work effort. To use RSA as an example, modern factoring algorithms can do much better than simply trying all the possible smaller prime numbers to factor a large composite. Diffie-Hellman is also subexponential. For com-



HASH ALGORITHM condenses a message into a digest, a digital fingerprint that can be used to detect forgeries. The text of the message is first converted into binary form. (The letter *A* might be represented by 00000, the letter *B* by 00001, the letter *C* by 00010, and so on.) The resulting string of 0s and 1s is then separated into equal-size blocks. Next, the chunks are fed in sequence as key material into a cipher. The final output is the digest, or hash, of the original message. Note that a message of any length will always yield a digest of fixed size. The operation is called "one way" because it is virtually impossible to recover a message from its hash. Also, the algorithm is designed so that any given two messages will almost certainly yield distinct hashes, and it is computationally infeasible to find another message that produces the same hash as a given message. Thus, a digest can serve as a "fingerprint" for its corresponding message.

parison's sake, a 3,000-bit RSA or Diffie-Hellman key requires about the same amount of work to crack as a 128-bit key for a block cipher.

Still, block ciphers are hardly invincible. This year a special-purpose massively parallel machine built for less than $250,000 by the Electronic Frontier Foundation, headquartered in San Francisco, broke a DES message by exhausting its 56-bit key space in less than a week.

Brute force is not the only way to crack a cipher. Cryptanalysts can apply powerful mathematical and statistical tools to find any shortcuts, perhaps by uncovering patterns in the encrypted text. Attempts to break ciphers can be grouped into three categories, depending on how much is known about the original message (called plaintext) and the corresponding enciphered transmission (called ciphertext).

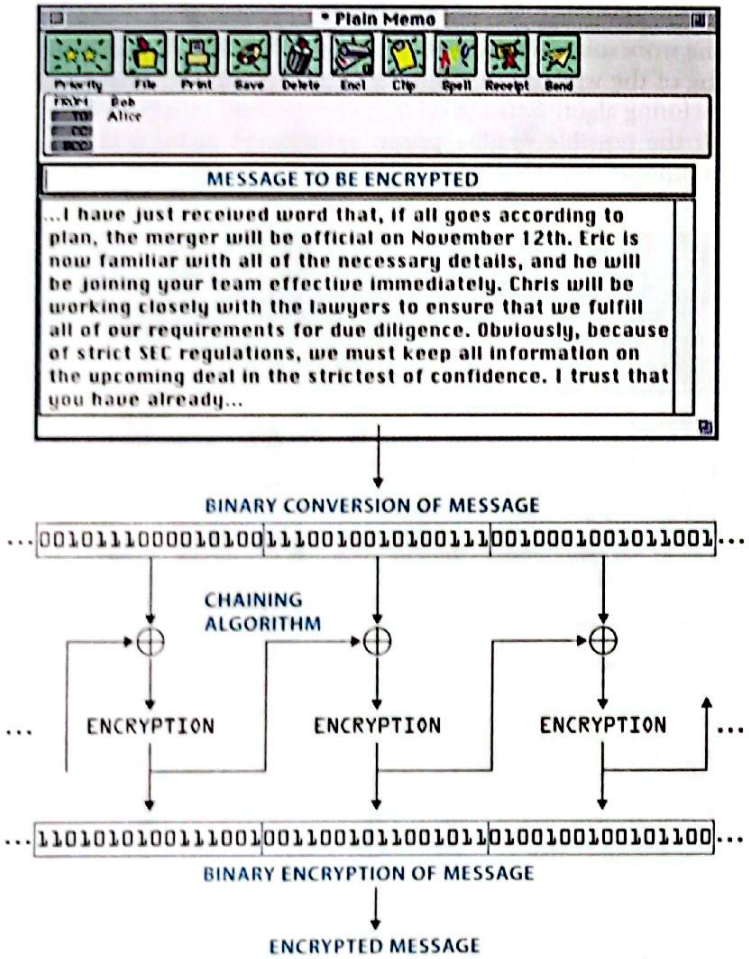
In some cases, all that the attackers have to work with is the ciphertext, so they have little to guide their efforts in guessing the key. Even a poorly designed cipher might be able to withstand such ciphertext-only attacks.

But if the attackers know at least a part of the message—for instance, that the text begins with "Dear Mr. Jones"—the opportunities for success increase significantly. At a minimum, they can try different keys until they find one that decrypts the "Dear Mr. Jones" part of the plaintext. Even if the attacker knows only the language (Russian or French or COBOL) of the plaintext, that information can be exploited. If the message is in English, for example, the most common word is probably "the." To thwart such known-plaintext attacks, some cryptography systems electronically compress the message, squeezing out easily predictable patterns in the plaintext, before encrypting it.

Often an attacker knows much more. If a person steals a "smart" card containing crypto hardware, the thief can present perhaps billions of carefully chosen messages to the card and study the ciphertext output. Such chosen-plaintext attacks will crack a poorly designed cipher easily. Another example is public-key systems. An attacker can write a message, encrypt it with the public key (which is, after all, public) and then analyze the resulting ciphertext.

Two very effective methods of cryptanalysis, differential and linear, have recently been developed. Both approaches have been used to crack a number of well-known block ciphers and to show that DES can be broken hundreds or thousands of times faster than by key exhaustion.

In differential cryptanalysis, introduced by Shamir and Eli Biham of Technion Israel Institute of Technology, many pairs of plaintext messages with carefully chosen differences are encrypted to find a corresponding pair of ciphertexts that have a certain dissimilarity. When such a pair is found, it reveals information about the key. Linear cryptanalysis, developed by Mitsuru Matsui of Mitsubishi Electric Corporation, searches for correlations between plaintext, ciphertext and key that are true slightly more often than not. The method then gathers statistics on large numbers of known plaintext-ciphertext pairs, looking for biases that will disclose clues about the key.



CHAINING ALGORITHM increases the security of block ciphers. A message is converted into a string of 0s and 1s, and the long sequence is then broken into blocks of equal size. Before each of these chunks is encrypted, it is first mathematically combined with the enciphered previous block. Thus, the encryption of the 23rd chunk depends on the enciphered 22nd block, which itself was affected by the encryption of the 21st block, and so on. Because of this feedback chain, an encrypted block depends on all the previous blocks, making the cipher more difficult for cryptanalysts to crack.

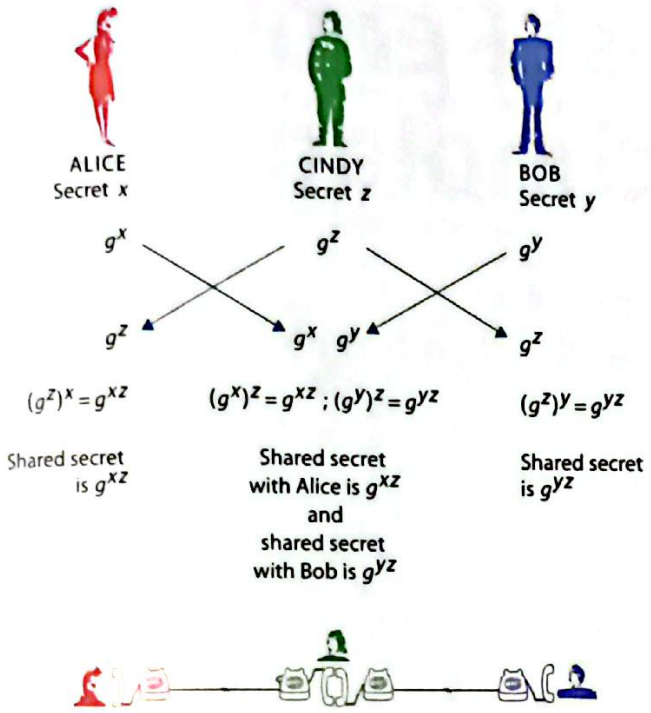
Beware of Middlemen

Though powerful, cryptanalysis techniques usually require a backbreaking number of computations. Often instead of trying to crack a cipher, it is easier to attack the protocol, or implementation, of that cipher.

One potential threat is man-in-the-middle attacks, which are the biggest vulnerability of public-key cryptosystems. When Bob wants to send a message to Alice, he may be unaware that Cindy is attempting to impersonate Alice. If Cindy can trick Bob into using her public key instead of Alice's, she will be able to decrypt Bob's message.

The only way to prevent this type of attack is for Bob to confirm somehow that Alice's public key is really Alice's. Most of the complexity of well-designed implementations of public-key cryptosystems is devoted to this one particular vulnerability. One solution is to have a trusted third party verify and sign the keys. This approach, however, begs the

ILLUSTRATIONS BY DMITRY KRASNY



DLEMAN ATTACK is the greatest vulnerability of public-key cryptems. If Cindy, an eavesdropper, can intercept transmissions between Alice and Bob, she can trick Bob into using her g^z instead of Al- g^x and similarly deceive Alice into using g^z instead of Bob's g^y [*see on page 84*]. Cindy would then be able to decrypt and reencrypt 's and Bob's messages to each other—all unbeknownst to the le. The process is analogous to Alice and Bob talking on special, pted telephones while Cindy listens in by using a pair of such es to decrypt, then reencrypt, the transmission.

or—and politically controversial—question: Should the s be certified in a top-down manner by government aurities or in a decentralized grassroots method by different entities, including various private companies and individuals, allowing people to choose for themselves which key signers to trust? In fact, this issue is so crucial that I could have written this entire article on it.

As cipher-breaking techniques have improved, so have the algorithms for stronger cryptography. Recently the National Institute of Standards and Technology solicited designs for the Advanced Encryption Standard (AES), a new block cipher to replace the DES, which has reached the end of its useful life, mainly because of its short 56-bit key and 64-bit block size. The AES, which has been generating considerable excitement in the cryptography field, will use a key size of 128, 192 or 256 bits to encrypt data in 128-bit blocks.

Good AES designs will meet several criteria. They will offer flexibility in various key and block sizes; they will be efficient in setting up keys and in encrypting and decrypting, particularly when implemented on 32-bit processors as well as on eight-bit microprocessors, such as in "smart" cards, and on other hardware; and they will perform well in a wide range of applications, from satellite communications to high-definition television.

Several of the AES candidates appear to be extremely well designed. The better proposals have capitalized on the experience of cryptographers who have studied block ciphers for the past 20 years, including their knowledge of how to defend against linear and differential cryptanalyses.

Of the 15 submissions, I believe more than a few would make credible encryption standards. MARS, which draws on the experience of IBM's original DES team, uses two very different structures for the encryption rounds. The mixed approach, the IBM cryptographers claim, will result in better security than that achieved with a homogeneous cipher. CAST-256 extends the earlier CAST architecture to a 256-bit key and 128-bit block size. Twofish is more mathematically rigorous than its predecessor, Blowfish. Serpent deploys an unusual parallel design to make it as fast as DES, with a short time for key setup, which should enable the cipher to be used efficiently as a hash function.

Deciphering the Future

Whichever candidate is selected, the AES promises to tip the balance further in favor of cryptographers in their ongoing arms race against cryptanalysts. Today the very best cryptosystems are beyond the reach of the best cryptanalytic methods known. Still, it is conceivable that powerful, new cipher-breaking techniques will be developed in the coming years. Even so, many cryptologists contend that the gap between cipher makers and cipher breakers will only widen.

I agree with that assertion, in part because of the active community of cryptographers in academia and the private sector, which has grown and matured to reach parity with military expertise in the field. Evidence of this was supplied by the recent declassification of the Skipjack cipher, which the NSA had developed in secrecy for the Clipper chip. A review by Technion's Biham, an academic cryptologist, revealed the algorithm to be less conservative, with a smaller margin of safety, than the best designs from academia. It appears that cryptography—like the Internet itself—has stepped from the dark shadows of the military into the bright sunshine of the free market.

The Author

PHILIP R. ZIMMERMANN is the author of Pretty Good Privacy (PGP) encryption software, for which he received the Chrysler Award for Innovation in Design. He is a software engineer with more than 20 years of experience in cryptography, data communications and real-time embedded systems. He has testified before Congress, urging the U.S. government to loosen its control of encryption technology. Selected in 1995 by *Time* magazine as one of the 50 most influential people on the Internet, Zimmermann is currently a senior fellow with Network Associates.

Further Reading

THE OFFICIAL PGP USER'S GUIDE. Philip R. Zimmermann. MIT Press, 1995.

APPLIED CRYPTOGRAPHY. Second edition. Bruce Schneier. John Wiley & Sons, 1996.

Additional information can be found at http://www.pgpi.com, http://www.pgpinternational.com, http://www.pgp.com/phil, http://csrc.nist.gov/encryption/, http://www.epic.org, http://www.eff.org and http://www.cdt.org on the World Wide Web.

The Case against Regulating Encryption Technology

One of the pioneers of computer security says the U.S. government should keep its hands off cryptography

by Ronald L. Rivest

The widespread use of cryptography is a necessary consequence of the information revolution. With the coming of electronic communications on computer networks, people need a way to ensure that conversations and transactions remain confidential. Cryptography provides a solution to this problem, but it has spawned a heated policy debate. U.S. government agencies want to restrict the use of data encryption because they fear that criminals and spies may use the technology to their own advantage.

Before the 1970s, cryptography was too complicated and too expensive for everyday use. Two inventions changed this picture dramatically: public-key cryptography and the microprocessor. The idea of using public and private encryption keys—first proposed in 1976 by electrical engineers and computer scientists Whitfield Diffie, Martin E. Hellman and Ralph C. Merkle—paved the way for the general use of strong cryptography, which scrambles messages so effectively that it would take many years of computer time to break the code. And the growing availability of fast microprocessors gave more and more computer users the ability to make the calculations necessary for this kind of encryption.

As strong cryptography became easily accessible in the late 1980s and early 1990s, two government agencies grew concerned about its widespread deployment. The National Security Agency (NSA), which monitors electronic communications around the globe, worried that it would be unable to decipher the encrypted messages of potential spies and terrorists. Similarly, the Federal Bureau of Investigation feared that criminals in the U.S. would use the encryption software to thwart surveillance of their voice or data communications. Over the past decade these agencies have pushed for government regulation of encryption technology and have favored the continuation of current restrictions on the export of strong encryption software.

The government's concern is that the "bad guys" will benefit from the new cryptographic technology. This is certainly possible—the sun shines on the evil as well as the good. But it is poor policy to clamp down indiscriminately on a technology merely because some criminals might be able to use it to their advantage. For example, any U.S. citizen can freely buy a pair of gloves, even though a burglar might use them to ransack a house without leaving fingerprints.

I rather like the glove analogy; let me expand on it a bit. Cryptography is a data-protection technology just as gloves are a hand-protection technology. Cryptography protects data from hackers, corporate spies and con artists, whereas gloves protect hands from cuts, scrapes, heat, cold and infection. The former can frustrate FBI wiretapping, and the latter can thwart FBI fingerprint analysis. Cryptography and gloves are both dirt-cheap and widely available. In fact, you can download good cryptographic software from the Internet for less than the price of a good pair of gloves.

Should the use of cryptography be restricted to satisfy the concerns of the NSA and the FBI? It is true that these two agencies may find their jobs more difficult as cryptographic technology spreads. But we should also consider cryptography's benefits to society as a whole. Most people use cryptography to prevent crime rather than to hide it, just as most people wear gloves to protect their hands rather than to hide their fingerprints. By ensuring the confidentiality and authenticity of electronic banking and Internet commerce, cryptography prevents theft and credit-card fraud. The vigorous application of cryptography may also improve national security: the encryption of communications, for example, protects U.S. businesses from industrial espionage. Paradoxically, we may create a safer society by promoting a technology that somewhat hampers law enforcement.

Some have hoped for compromise solutions that would allow strong cryptography to be widely used while still enabling the NSA and the FBI to decrypt messages when lawfully authorized to do so. For example, there have been key-escrow proposals that would require users to register their software encryption keys with law-enforcement agencies, and key-recovery proposals that would give government agencies backdoor access to the keys. In a typical key-recovery scheme, an encrypted version of the message encryption key is sent along with each message. An FBI-authorized key-recovery center can use a master backdoor key to decrypt the message key, which is then used to decrypt the message itself.

In my opinion, these systems would satisfy no one. They are very easy to circumvent: spies and criminals could modify the encryption software to disable the key-recovery features, or they could simply download alternative software

from the Internet. Key recovery would be very expensive, too. Someone would have to pay for creating, staffing and maintaining the key-recovery centers. But the most subtle and serious cost in the long run would be the erosion of confidence in the government resulting from an increased sense of "Big Brotherism." To get an idea of the intrusiveness and impracticality of key recovery, imagine that whenever you bought a pair of gloves you were legally required to sew latex copies of your fingerprints onto the gloves' fingertips!

SLIM FILMS

Key-recovery systems would also create substantial security risks. The system's most serious flaw is that the same back doors used by the FBI to decipher encrypted messages would become targets for criminals, hackers, spies and even disgruntled employees of the FBI itself. If criminals or hackers managed to penetrate a key-recovery center and steal a master backdoor encryption key, they would be able to decrypt Internet communications at will. Millions of corporate, personal and government secrets would suddenly become vulnerable to theft and tampering.

In 1993 Congress asked the National Research Council to study U.S. cryptographic policy. The council then convened a blue-ribbon committee of 16 members. Its superb 1996 report, the result of two years' work, offered the following conclusions and recommendations:

- "On balance, the advantages of more widespread use of cryptography outweigh the disadvantages."
- "No law should bar the manufacture, sale or use of any form of encryption within the United States."
- "Export controls on cryptography should be progressively relaxed but not eliminated."

The committee members concluded that a ban on unregulated encryption would be "largely unenforceable." But the FBI and the NSA continue to push for key recovery and to oppose the relaxation of export controls unless key recovery is incorporated into the exported software.

Strong cryptography only gets easier to implement—and harder to regulate—over time. Professional societies are adopting public cryptographic standards that even a high school student can convert into programs. And new techniques such as "chaffing and winnowing"—which does not encrypt a message but achieves confidentiality by hiding pieces of the message in a welter of random data, or chaff—illustrate the enormous technical difficulties involved in trying to control cryptography.

The economic consequences of our current policy are also becoming clearer. A recent study conducted by the Economic Strategy Institute, a think tank in Washington, D.C., concluded that continuing the export controls on cryptographic products will cost the U.S. economy more than $35 billion over the next five years. My personal opinion is that the U.S. risks losing its leadership position in the software industry because of its restrictive export policy.

Finally, the ability to have private conversations is in my view an essential democratic right. Democracy depends on the ability of citizens to share their ideas freely, without fear of monitoring or reprisal; this principle should be upheld as much in cyberspace as it is in the real world. For the U.S. to restrict the right to use cryptography would be a setback for democracy—and a victory for Big Brother.

The Author

RONALD L. RIVEST is the co-inventor of RSA encryption, the most widely used public-key cryptosystem. He is Edwin S. Webster Professor of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology and a founder of RSA Data Security (a subsidiary of Security Dynamics Technologies).

Further Reading

CRYPTOGRAPHY'S ROLE IN SECURING THE INFORMATION SOCIETY. Edited by Kenneth W. Dam and Herbert S. Lin, National Research Council. National Academy Press, 1996. The report can be found at http://www.nap.edu/readingroom/books/crisis on the World Wide Web.

THE ELECTRONIC PRIVACY PAPERS: DOCUMENTS ON THE BATTLE FOR PRIVACY IN THE AGE OF SURVEILLANCE. Bruce Schneier and David Banisar. John Wiley & Sons, 1997.

PRIVACY ON THE LINE: THE POLITICS OF WIRETAPPING AND ENCRYPTION. Whitfield Diffie and Susan Landau. MIT Press, 1998.