12 The dark side of the web

When he later connected the same laptop to the Internet, the worm broke free and began replicating itself, a step its designers never anticipated.
David E. Sanger[1]

Black hats and white hats

As we have seen in Chapter 10, the Internet was invented by the academic research community and originally connected only a relatively small number of university computers. What is remarkable is that this research project has turned into a global infrastructure that has scaled from thousands of researchers to billions of people with no technical background. However, some of the problems that plague today's Internet originate from decisions taken by the original Internet Engineering Task Force (IETF). This was a small group of researchers who debated and decided Internet standards in a truly collegial and academic fashion. For a network connecting a community of like-minded friends and with a culture of trust between the universities, this was an acceptable process. However, as the Internet has grown to include many different types of communities and cultures it is now clear that such a trusting approach was misplaced.

One example is the IETF's definition of the Simple Mail Transfer Protocol (SMTP) for sending and receiving email over the Internet. Unfortunately, the original SMTP protocol did not check that the sender's actual Internet address was what the email packet header claimed it to be. This allows the possibility of *spoofing*, the creation of Internet Protocol (IP) packets with either a forged source address or using an unauthorized IP address. Such spoofing is now widely used to mask the source of cyberattacks over the Internet, both by criminal gangs as well as by governments.

It is difficult to predict the consequences of any new technology. Along with benefits there are often some downsides that later emerge. One such downside was the emergence of *spam* emails. Spam consists of unsolicited commercial emails that are now sent out to millions of email users in a bulk mailing. The email spam costs the spammer very little to send and even if only a tiny percentage of recipients respond it can be a very profitable business. One of the first spam emails was sent to the ARPANET community by

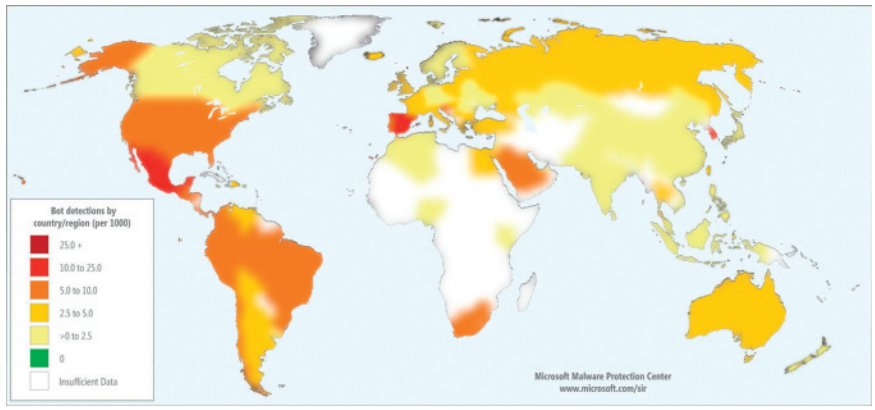


Fig. 12.1. An example of worldwide botnet detections by the Microsoft Digital Crimes Unit.

an overenthusiastic DEC marketing representative in 1978. Since then, the volume of email spam has grown enormously. A 2003 study estimated that more than half the email transmitted over the Internet was spam and that more than 90 percent of all spam email was sent by just 150 people. By 2011, according to one estimate, spam emails accounted for more than 80 percent of all email sent over the Internet. Increasingly, these spam emails are not sent by identifiable individuals but by *zombie computers* or *botnets* (Fig. 12.1) as we discuss in the following text. Botnets are made up of personal computers belonging to ordinary users whose machines have been taken over by computer malware that can be instructed to send out spam. Fortunately, *spam filters* are now available that can identify most spam emails and redirect them straight to the "junk" email folder.

Malware is short for *malicious software* and means software that is designed to gain unauthorized access to computers for a range of purposes, some relatively harmless and others definitely criminal. The popularity of the Unix operating system in universities and businesses in the 1970s and 1980s originally made Unix a prime target for *black hat* hackers, clever programmers who use their skills to gain unauthorized access to computer files. Nowadays, because of the success of the personal computer, Microsoft Windows is the operating system most under attack. In many cases, the hackers are able to gain control of the high-level system security privileges of the *system administrators*, the people responsible for keeping the computer system running. On the other side in this hacking war are the *white hat* hackers. These are ethical computer security experts who specialize in finding security loopholes and in defending computer systems from cyberattacks.

The techniques used by the black hats are many and varied. We begin by discussing a selection of the most common techniques before looking at the recent escalation in the use of malware for cyberwarfare. We then take a brief look at modern cryptographic systems that are designed to keep Internet communications secure from eavesdroppers and end with some comments on cookies, spyware, and privacy.

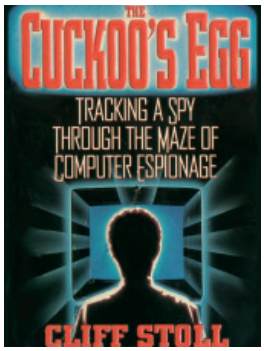


Fig. 12.2. A fascinating detective story about Cliff Stoll chasing a hacker during the ARPANET era.

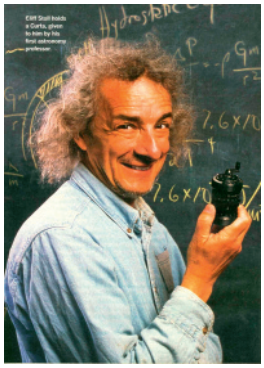
Cyberespionage

Clifford Stoll's (B.12.1) classic book *The Cuckoo's Egg* describes the complexity of tracking and prosecuting a black hat hacker (Fig. 12.2). Stoll was an astronomer turned system administrator for the computers at Lawrence Berkeley National Laboratory. The lab's computers ran Berkeley Unix and had two systems of accounting software for keeping track of the usage of these machines – one a standard Unix utility program and the other a homegrown program specific to Berkeley. From a seventy-five-cent discrepancy in the computer accounts at Lawrence Berkeley National Laboratory in 1986, Stoll deduced that someone was hacking into the lab's system. By sleeping in the lab and being alerted to every incoming computer connection, Stoll was able to record the exact keystrokes that had been used when the offense occurred. The results were surprising.

The hacker had gained access to one of Stoll's computers by guessing the password for an old, inactive user's account. When in the system, he then used a bug in the popular GNU-Emacs editor program to trick the computer into giving him the same privileges as a system administrator, so-called super-user or root privileges. This bug allowed him to move a file from his user area into what should have been an area of memory restricted to the system manager. The GNU software did not check whether the area was in the protected system software memory space. Once in this privileged area, the hacker then ran a counterfeit version of a standard Unix program, *atrun*, which runs queued up jobs at regular intervals. This unauthorized program is the cuckoo's egg of the title of the book – named for the cuckoo's trick of laying its eggs in nests of other birds. Running the counterfeit program allowed the hacker to gain the super-user privileges of a system administrator. He then restored the real Unix atrun program and erased his tracks from the system log so that the systems administrators would see nothing wrong. He also scanned all email messages for references to "hacker" and "security" and used his new privileges to kill the program of any user who he thought might have been monitoring his activity.

The situation was extremely serious: the hacker could read anyone's email, access or delete any file, and set up a new, hidden account that could provide him with a "backdoor" into the computer known only to him. All of the data stored on the computer was now at risk. Moreover, from his position as super-user, he was able to explore not only all the other computers at the Berkeley Lab connected by the Local Area Network (LAN), but also the computer systems connected to Berkeley through the ARPANET.

Stoll watched the hacker systematically attempting to break into several military computer installations by guessing passwords or by using unprotected guest or visitor accounts. It was surprising how many supposedly secure military sites still used the standard factory password settings for their super-user system administrator accounts. After a long chase – and remarkable initial indifference from the Federal Bureau of Investigation (FBI), the Central Intelligence Agency, and even the National Security Agency (NSA) – the trail led to West Germany (Fig. 12.3). The hacker, Markus Hess, was part of a group selling sensitive information obtained from these U.S. military computing systems to the Soviet Union.



B.12.1. Clifford Stoll is a U.S. astronomer and author who is probably best known for his book *The Cuckoo's Egg*. This tracked a hacker who had broken into Stoll's computer at Lawrence Berkeley Laboratory back to Hanover, Germany.

Fig. 12.3. The NSA was established in 1952 to handle secret communications and gather intelligence.

The Berkeley hacker used another technique to steal passwords: he had installed a *Trojan horse* program. In Virgil's *Aeneid*, when the Greeks pretended to abandon their siege of the city of Troy, they left behind a giant wooden horse. The citizens of Troy took the horse into the city and celebrated the defeat of the Greeks. In fact, the horse was full of Greek soldiers and the Trojans had brought the enemy inside their defenses, leading to the sacking of their city. A Trojan horse program does much the same thing for a computer system. It hides malicious or harmful code inside an apparently harmless program so that it can get control and do damage. At Berkeley, the hacker produced his own version of the standard login program to capture users' passwords. A would-be user was greeted by what looked like the normal login message:

WELCOME TO THE LBL UNIX-4 COMPUTER
PLEASE LOGIN NOW
Login:

After the user typed the account name, the system then asked for the password:

ENTER YOUR PASSWORD:

The user entered the password, which was copied along with the account name into a file set up by the hacker. The program then responded:

SORRY, TRY AGAIN

The user is then returned to the real login page and logs in as usual, unaware that the account details and password have been stolen. Such Trojan horse techniques are now widely used to capture private personal information and bank account details.

Viruses, rootkits, and worms

In principle, the damage caused by a Trojan horse program is restricted to one computer. A computer *virus*, as the name implies, is nastier in that it is designed to spread to other computers. The code for a virus is a small set of instructions incorporated into an application rather than a complete, stand-alone program. Initially, computer viruses were spread by the exchange of infected floppy disks but are now more typically spread using the Internet by getting users to click on harmless-looking email attachments like a photograph or a document. One of the first major virus attacks was the "Brain" virus (Fig. 12.4). Two Pakistani brothers created it in 1986, targeting bootable floppy disks for PCs running MS-DOS. A *bootable floppy disk* was one that held its own operating system and was usually used to restart a failed system or to install a new operating system. When the PC was *booted* (started up) from the infected disk, the computer loaded the Brain virus before executing the original MS-DOS code. The virus hid itself from the user by reporting the sectors of the floppy disk on which it was installed as damaged. If the user actually checked the boot code on the disk, the original uninfected code would be displayed rather than the modified code including the virus. In this case, the result was relatively harmless: the virus spread an advertisement for the brothers' company with its name and contact details, a genuine example of "viral" advertising.

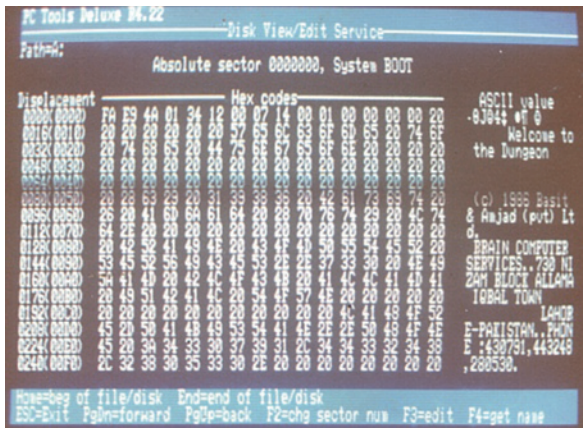


Fig. 12.4. A screenshot of the BRAIN virus in 1984; it was one of the first PC viruses.

After the example of the Brain virus, hackers developed many thousands of new viruses, often using clever new techniques to help them spread. One of the most striking was produced in Germany in 1987. It was called the Cascade virus because it made the characters on the screen appear to fall to the bottom. This virus also introduced a new level of sophistication by using *encryption* techniques, which convert messages into secret code, to hide the details of its internal workings. We will discuss encryption later in this chapter. It was this explosion of computer viruses in the 1990s that led to the creation of a whole new industry – with antivirus companies now providing software to combat malware.

As a footnote, the term *computer virus* was probably first used by Len Adleman, a professor at the University of Southern California, well known for his contributions to cryptography. His student Fred Cohen was studying computer infections and defined a virus as "a computer program that can affect other computer programs by modifying them in such a way as to include a (possibly evolved) copy of itself."[2] In November 1983, Cohen demonstrated a computer virus that infected the Unix file directory program. After some other experiments with program infections, Cohen examined the theoretical difficulty of detecting computer viruses. His PhD thesis in 1986 showed that there is no way of definitively detecting a virus. The best we can do is to assemble a collection of tricks and informal techniques, sometimes known as *heuristics*, to supplement our guesswork.

The Brain virus was one of the first to use *cloaking* techniques to hide the program from common system administrator and diagnostic utilities. In Unix, the traditional name for the most privileged account is *root*, and software designed to give a user root privileges is sometimes known as a *rootkit*. The term *rootkit* is now applied more generally to types of malware that use cloaking techniques to make themselves invisible to antivirus software and standard system tools. Rootkits came to prominence in 2005 when the Sony BMG music group installed overaggressive copy protection measures on twenty million music CDs. When the CD was used, it secretly installed software that actually modified the operating system to prevent CD copying. Moreover, the software was very difficult to remove and used the same rootkit cloaking techniques as conventional malware to hide its presence. The scandal came to light when security researcher Mark Russinovich (B.12.2) posted a detailed technical

B.12.2. Mark Russinovich was a security researcher at his Winternals company when he became a victim of Sony BMG's CD rootkit. His subsequent blog post on the technical aspects of the rootkit showed how it installed itself and modified the operating system of an unsuspecting user. Russinovich is now a Technical Fellow at Microsoft and the author of the novels *Zero Day* and *Trojan Horse*.

B.12.3. Robert Morris Sr. (1932–2011) was chief scientist of the NSA's National Computer Security Center at the time of Clifford Stoll's cuckoo's egg experiences with cyberespionage. Before he joined the NSA in 1986, Morris had been a researcher at Bell Labs working on both the Multics and Unix operating systems.

description of the Sony rootkit on his blog in October 2005. He also discovered that the software created new security loopholes and could lead to system crashes. Sony BMG's reaction to this revelation was initially: "Most people don't even know what a rootkit is so why should they care about it?"[3] However, the company eventually recalled and replaced the affected CDs and abandoned its extended copyright protection software. Mikko Hypponen, chief research officer at the Finnish-based security company F-Secure, commented:

> [The] Sony rootkit was one of the seminal moments in malware history. Not only did it bring rootkits into public knowledge, it also gave a good lesson to media companies on how not to do their DRM [digital rights management] solutions.[4]

The term *computer worm* is generally used to describe malware that is designed to spread from computer to computer but, unlike a virus, which must attach itself to a program or file to spread, a worm is a complete program capable of replicating all by itself. At Xerox PARC in 1978, John Shoch was experimenting with a program that could seek out Alto machines on the Ethernet that were not being used, boot up the machine to do some work, and replicate by sending copies of itself to other idle machines on the network. One of his experiments went wrong and, after leaving his program running overnight, Shoch was awakened by angry users complaining that he had crashed their Altos. Eradicating the worm proved very difficult, and it was fortunate that he had equipped his worm program with a "suicide capsule" that he was able to activate. Shoch called his program a *worm*, after the idea of the "Tapeworm," software that runs by itself in John Brunner's science fiction novel *The Shockwave Rider*.

Worms came into public prominence through the "Internet worm" attack on the ARPANET in 1988. Clifford Stoll, then at Harvard, described this "Internet worm" attack in graphic detail:

> As fast as I'd kill one program, another would take its place. I stomped them all out at once: not a minute later, one reappeared. Within three minutes there were a dozen.[5]

B.12.4. Robert Morris Jr. was a graduate student at Cornell when he created the first worm on the ARPANET in 1988. He was the first person to be convicted under the USA Computer Fraud and Abuse Act. He is now a tenured professor at MIT.

Stoll informed Bob Morris (B.12.3), chief scientist at the NSA, whom he knew from his investigation of the Berkeley hacker, of the ongoing worm attack. Stoll was not amused to be called back a few hours later by someone from the NSA who asked if he was the person who had written the worm program! While other ARPANET node system administrators across the United States were decrypting the worm program, Stoll tracked down the place where the worm had been released. By a supreme irony, the trail led back to Bob Morris Jr. (B.12.4), a graduate student at Cornell University and the son of Bob Morris Sr. of the NSA. The Morris worm was not the first worm program, but it was certainly one of the most damaging. Stoll estimated that it infected two thousand machines within fifteen hours.

Morris's worm was a significant escalation in malware for two reasons. First of all, the program automated all sorts of tricks that a hacker might use in attempting to break into a computer system. Given access to one computer, the worm would first check if it was automatically given privileges to run programs

Fig. 12.5. The infamous Morris worm was only a short C program, yet it shut down large portions of the ARPANET in November 1988.

on other computers; then it would try a long list of common passwords. If these attempts failed, it would then try some other vulnerability, such as a flaw in the Unix Sendmail program, well known to computer experts at the NSA. The second reason for its importance was that if all these attempts failed, Morris had exploited a new type of bug called *buffer overflow*. The Unix operating system is written in the C programming language, and the first book about C was written by Bell Labs researchers Brian Kernighan and Dennis Ritchie. The book shows how to write a program to read a series of input characters into computer memory using an area of memory called a *buffer*. In their example code, the size of the buffer was specified but not whether the number of characters being entered actually exceeded this size. The younger Morris realized that the extra characters would overwrite the rest of the program's data and instructions. By placing judicious machine instructions in these overflow characters, a hacker could use this flaw to gain the root privileges of a super-user. Morris also encrypted the virus software to make it more difficult to find out what the program did and also used several techniques to avoid detection. The worm infected thousands of computers, and system managers took several days to disinfect their computers. Morris was convicted of a felony in May 1990 and sentenced to three years of probation, four hundred hours of community service, and a $10,000 fine (Fig. 12.5).

The story had a happy ending for Morris. After his conviction, Xerox PARC invited him to become an intern student there and he is now a professor at MIT. However, an unfortunate outcome of Morris's worm was that it demonstrated a new way of attacking computers. Such unchecked memory buffers occurred in almost all Unix programs and also in Windows. After a hacker called "Aleph One" put up a detailed "instruction manual" (Fig. 12.6) on the Web in 1996, buffer overflow became a relatively straightforward technique for black hats to adapt. In 1992, there were estimated to be around 1,300 viruses or worms; in 1996, more than 10,000; and by 2002, more than 70,000. By 2003, the Slammer worm had set a record for spreading faster than any previous malware, infecting seventy-five thousand computers in just ten minutes.

Fig. 12.6. Aleph One's paper on the buffer overflow vulnerability.

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

File 14 of 16

BugTraq, r00t, and Underground.Org
bring you

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Smashing The Stack For Fun And Profit
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

by Aleph One
aleph1@underground.org

`smash the stack` [C programming] n. On many C implementations it is possible to corrupt the execution stack by writing past the end of an array declared auto in a routine. Code that does this is said to smash the stack, and can cause return from the routine to jump to a random address. This can produce some of the most insidious data-dependent bugs known to mankind. Variants include trash the stack, scribble the stack, mangle the stack; the term mung the stack is not used, as this is never done intentionally. See spam; see also alias bug, fandango on core, memory leak, precedence lossage, overrun screw.

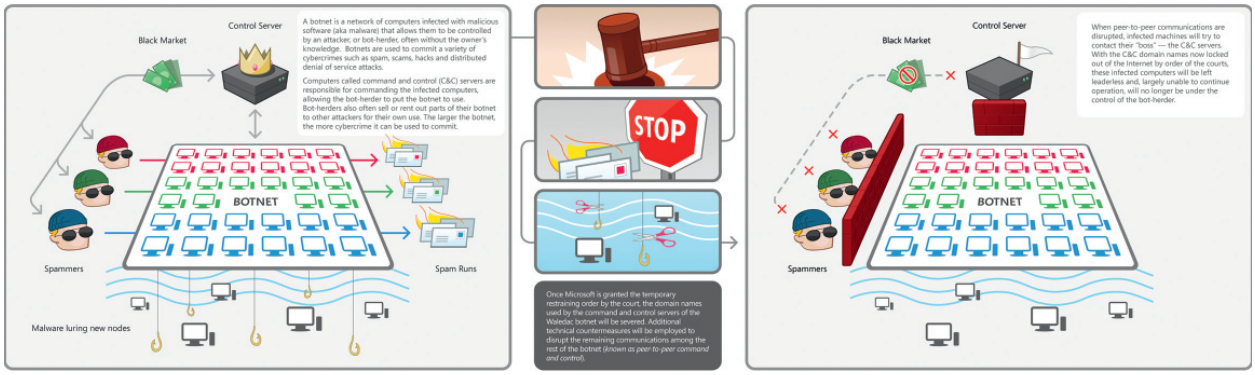


Fig. 12.7. Nefarious botnet programs work by hijacking millions of computers, usually without their owners' knowledge.

Botnets and zombie computers

The last decade has seen a dramatic rise of hacking for profit by criminal organizations. *Botnets* are collections of computers that have been taken over by techniques such as those described above and are controlled by so-called *bot-herders* (see Fig. 12.7). *Bot* is short for *robot program*, and sometimes these enslaved computers are known as *zombie computers*. The botnets can be used to conduct *denial of service attacks* on specific websites (see Fig. 12.8) – such attacks try to shut down a site by bombarding it with so many requests that the system is forced to shut down, denying service to legitimate users. Botnets can also be used to send spam or to capture personal details by *key logging* – capturing a user's keystrokes. A recent example is the Conficker botnet that first appeared in 2008. It was estimated to have infected more than ten million computers around the world and to have the capacity to send an incredible ten billion spam emails per day. Mark Bowden's book, *Worm*, details how the white hat security community collaborated with Microsoft to contain and partially eliminate the threat to the Internet posed by Conficker. However, a 2012 Microsoft report states that

> … the Conficker worm was detected approximately 220 million times worldwide in the past two and a half years, making it one of the biggest ongoing threats to enterprises. The study also revealed the worm continues to spread because of weak or stolen passwords and vulnerabilities for which a security update exists.[6]

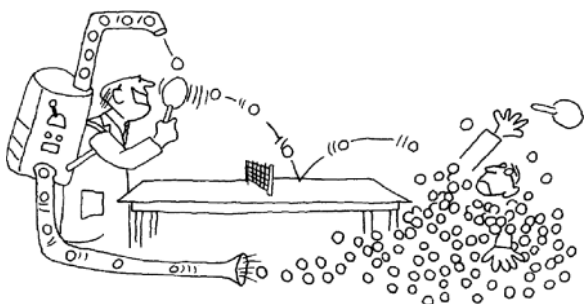


Fig. 12.8. This cartoon with the original caption "… filibustering destroys communication" captures the essence of a "denial of service" attack.

In another recent example, the Microsoft Digital Crimes Unit worked with the financial industry and the FBI "to disrupt more than 1,400 Citadel botnets which are responsible for over half a billion dollars in losses to people and businesses."[7]

A last example is particularly worrying. In the case of the Nitol botnet (Fig. 12.9), Microsoft found that nearly 20 percent of brand new PCs purchased through unsecure Chinese supply chains were already preinfected with Nitol malware.

> A supply chain between a manufacturer and a consumer becomes unsecure when a distributor or reseller receives or sells products from unknown or unauthorized sources. In Operation b70, we discovered that retailers were selling computers loaded with counterfeit versions of Windows software embedded with harmful malware.[8]

This malware is particularly worrisome since it can be spread to friends and colleagues through a USB memory stick.

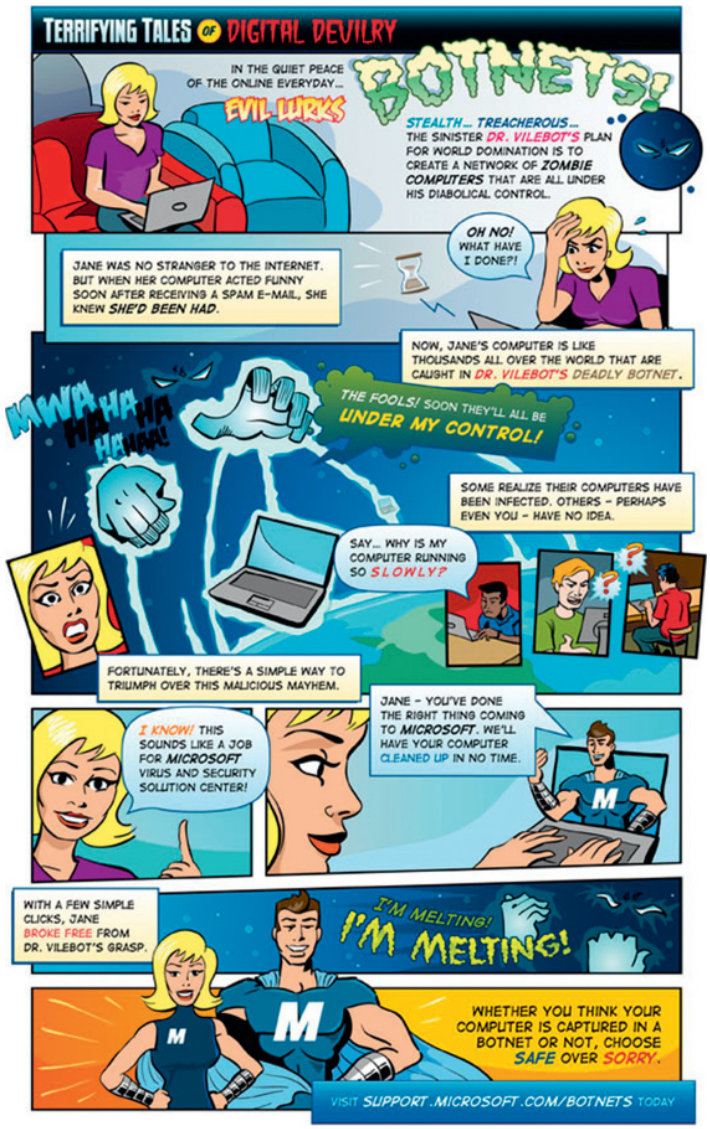


Fig. 12.9. A cartoon strip from Microsoft showing how to evade evil botnets.

Cyberwarfare

The latest escalation in malware is the potential to use worms for *cyberwarfare*, politically motivated hacking for purposes of spying or sabotage. It is now believed that the Stuxnet worm discovered in the summer of 2010 was engineered by U.S. and Israeli computer experts specifically to attack centrifuges at the Iranian uranium fuel-enrichment plant in Natanz, a site suspected of being a center for building a uranium-based atomic bomb. A series of high-speed centrifuges is needed to separate the rare, bomb-grade uranium-235 isotope from the much more common uranium-238 isotope present in uranium ore. An industrial control system manufactured by Siemens AG manages the centrifuges at the Natanz plant. This system uses a special-purpose computer called a *programmable logic controller* (PLC) that is programmed using Siemens software called Step-7. To spread itself throughout the plant, the Stuxnet worm exploited several previously unknown bugs – known as *zero day* bugs – in the Microsoft Windows XP operating system. In this way, the worm seized control of the PCs and substituted its own version of Siemens's Step-7 PLC code. This code modified the operation of the centrifuges yet reported to the operator that everything was fine. Thus the Step-7 malware used rootkit techniques to conceal its presence. The writers of worm code needed a very deep knowledge of both Windows and Siemens industrial control systems as well as detailed information about the centrifuge installation at the Natanz fuel-enrichment plant. It is likely that the worm was introduced into the Natanz using a USB memory stick since the plant is believed to be *air-gapped* – not connected to the Internet. A recent book, *Confront and Conceal*, by New York Times reporter David Sanger, describes operation *Olympic Games*, the codename for Stuxnet development and deployment, and details how the worm escaped to the Internet.

How much damage did Stuxnet cause? One report suggests that as many as a thousand centrifuges at Natanz or around 10 percent of the total needed to be replaced. In the long run, what may be of more significance than the cyberattack on Natanz is that the Stuxnet worm represents a blueprint for the construction of malware capable of attacking a wide range of industrial control systems, which form a key part of the modern world's critical infrastructure.

Fig. 12.10. The word *cryptos* (κρυπτός) in Greek means hidden. This is the Kryptos sculpture located in front of CIA headquarter in Langley. There are four messages on the sculpture; three of them have been deciphered, the fourth is so far unbroken.

Cryptography and the key distribution problem

The science of cryptography dates back to ancient times (Fig. 12.10). It consists of techniques for *encoding* the information in a message – that is, for putting the information into a form that can only be read or *decoded* by the intended recipient. According to the Roman historian Suetonius, Julius Caesar used a method called a *shift cipher* to encode secret government messages:

> If he had anything confidential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet, that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely D, for A, and so with the others.[9]

Fig. 12.11. Cipher disk invented by the Renaissance artist Leon Battista Alberti (1404–72). The brass inner disk can be rotated to align its letters with the letters of the outer circle.

A shift cipher consists of a *key* or number, known only to sender and receiver, which tells you how far to shift a second alphabet that is written under the first one (Fig. 12.11). Alan Turing, with help from Polish intelligence and others at Bletchley Park in the United Kingdom, built one of the first primitive computers to break the German Navy's Enigma codes. To break the German High Command's more complex Lorenz codes, Tommy Flowers built Colossus, arguably the first serious digital computer. With the advent of modern computers, cryptographers no longer needed to rely on a mechanical cipher machine to do the encryption and decryption. A computer can do the work of a complex cipher machine and still operate many times faster than any mechanical device. Since computers operate on binary numbers, messages must first be converted into a series of 1s and 0s according to some convention. There are now standard ways to encode characters and words into binary numbers. Once the message has been converted into a string of bits, encryption proceeds by scrambling the bits according to a method specified by a *key* that is shared by sender and receiver.

There are two main classes of cryptosystems, which are distinguished by whether the encryption key is shared in secret or in public. Gilbert Vernam (B.12.5) of AT&T proposed the one-time secret-key system in 1918. It is the only cryptosystem that provides absolute security. However, the system requires a key that is as long as the message and the keys must never be reused to send another message. Spies received a fresh set of keys in the form of a tear-off pad. After sending a message, the sender tore off the sheet with the used key and destroyed it. For this reason, the system is sometimes known as a *one-time pad*. When the Bolivian army captured the Marxist revolutionary Che Guevara in 1967, they found he had a list of random numbers that allowed him to send secret messages to Fidel Castro in Cuba. Guevara could do this securely over any radio link because he and Castro were using Vernam's one-time pad system.

B.12.5. Gilbert Vernam (1890–1960) came up with the idea of unbreakable encryption using so-called *one-time pads* of secret code numbers. The system was used by Che Guevara to communicate with Fidel Castro in Latin America.

In cryptology, the three participants in any discussion of coded messages are traditionally Alice, Bob, and Eve. Alice is the sender who wants to encrypt a message and send it securely to Bob. Bob is the receiver, who gets the message and wants to decrypt it and discover its meaning. Eve is a potential eavesdropper who wants to listen in and break the code. The one-time pad is secure because Alice encrypts the message using a random number as long as the message. Bob has the same key and can easily decrypt the message. The particular random number is only used once. Although this system is perfectly secure in principle, its weakness in practice lies in the fact that Alice and Bob have to share the same key and, since the keys are used only once, they need a great deal of them. The keys have to be distributed to Alice and Bob using some secure method, such as delivery by courier or a personal meeting. During World War II, the Russians foolishly reissued some one-time pads. This carelessness allowed U.S. cryptanalysts to decrypt a large number of previously undecipherable messages that they had intercepted over the years. This large-scale decoding effort was code-named the Venona project (Fig. 12.12). It was transcripts from this project that identified the atomic spy code-named CHARLES as the Los Alamos physicist Klaus Fuchs.

The weakness of secret-key encryption is the problem of distributing the keys safely. During World War II, the German military had to distribute

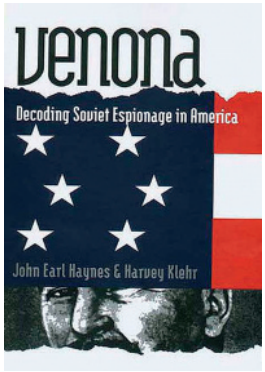


Fig. 12.12. The Venona project was a U.S. counterintelligence program to decrypt messages sent by the Soviet Union's intelligence agencies. The secret program was operational for more than forty years. Its existence was only revealed in 1995 after the end of the Cold War. The program identified Klaus Fuchs as the Manhattan Project spy who gave the plans for the atomic bomb to Stalin.

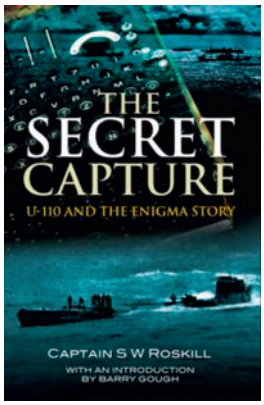


Fig. 12.13. *The Secret Capture* tells the story of how the British destroyer HMS Bulldog captured the German submarine U-110. The British sailors were able to retrieve the codebooks and an Enigma machine from the submarine and these were sent to the code breakers at Bletchley Park.

books containing a month's supply of keys to each operator of the Enigma code machine. For the U-boat fleet operating in the North Atlantic Ocean, this was a major logistical challenge and also a critical vulnerability. Ian Fleming, creator of James Bond, was a member of the United Kingdom's Naval Intelligence Division during the war. He suggested a James Bond–style plan called "Operation Ruthless" to capture the Enigma codebooks from a German ship. Although this particular operation was never carried out, the Allies did manage to capture intact Enigma codebooks from German weather ships and U-boats, enabling them to learn the locations of the Atlantic U-boat packs (Fig. 12.13).

The United States adopted the Data Encryption Standard (DES), a standard method of coding messages, in 1976. The DES was based on a system devised by the German-born cryptographer Horst Feistel, working at IBM's Thomas J. Watson Research Center in New York. It is widely believed that the U.S. government only allowed 56-bit keys so that the DES system was safe enough for normal users but not impossible for the NSA to break. Banks who needed to send secure messages of detailed transactions to each other were major users of encryption. To solve the problem of key distribution, banks employed dispatch riders who had to be thoroughly investigated and then equipped with padlocked briefcases. The costs of maintaining such a system rapidly became a major expense.

Diffie-Hellman key exchange and one-way functions

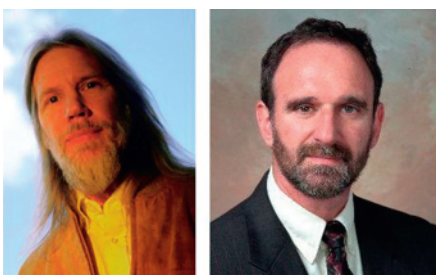
The way out of all these problems was to find a way for Alice and Bob to agree on a secret key without ever having to meet, in spite of Eve trying to listen in and discover the key. Remarkably, in 1976, agreeing on a secret key without meeting was shown to be possible. In his wonderful account of ciphers and cryptography, *The Code Book*, Simon Singh says of this new method of exchanging keys, "It is one of the most counterintuitive discoveries in the history of science"[10] and adds, "This breakthrough is considered to be the greatest cryptographic achievement since the invention of the monoalphabetic cipher, over two thousand years ago."[11]

The system that allows Alice and Bob to establish a secret key through a public discussion is called the *Diffie-Hellman key exchange*, after the inventors Whitfield Diffie and Martin Hellman (B.12.6). Hellman was a professor at Stanford University, and Diffie enrolled as his graduate student so they could both study the key distribution problem. Diffie and Hellman had realized that the solution to the problem required the use of a mathematical relationship called a *one-way function*. A two-way mathematical function is reversible in that it is easy to undo; a one-way function, as the name implies, is easy to do but very difficult to undo. Singh gives the following analogy: "Mixing yellow and blue paint to make green paint is a one-way function because it is easy to mix the paint but impossible to unmix it."[12] We can use this paint-mixing analogy to explain how Alice and Bob can establish a secret key without Eve finding out, even though she is able to monitor their public exchanges. We assume that each of the participants has a pot of yellow paint, and Alice and Bob each have another pot with their own secret color. They proceed as follows:

> If Alice and Bob want to agree on a secret key, each of them adds one liter of their own secret color to their own pot of yellow paint. Alice might add a peculiar shade of purple, while Bob might add crimson. Each sends their own mixed pot to the other and we assume that Eve can see and even sample these mixtures as they are sent between Alice and Bob. Finally, Alice takes Bob's mixture and adds one liter of her own secret color, and Bob takes Alice's mixture and adds one liter of his own secret color. Both pots should now be the same color, because they both contain one liter of yellow, one liter of purple and one liter of crimson. It is the exact color of the doubly contaminated pots that is used as the key.
>
> Does Eve know the secret key? No, she doesn't. She saw (and possibly sampled) the two partial mixtures that passed by her: "yellow and purple" and "yellow and crimson." If Eve combines these mixtures – the only operation she could do on her own – she will only end up with a mixture containing "yellow and yellow and purple and crimson." In order to find the secret key she would need to remove or "unmix" one unit of yellow from this mixture. Since she cannot unmix one unit of yellow she cannot generate the same color as Alice and Bob and thus does not know the key.[13]

So although Eve can intercept the pots of paint being exchanged, she cannot work out Alice's and Bob's secret keys because mixing paint is a one-way function.

The actual mathematical one-way function used in the Diffie and Hellman key exchange proposal was based on *modular arithmetic*. Calculations in modular arithmetic are done with a count that resets itself to zero every time a certain number, known as the *modulus*, has been reached. Modular arithmetic is like telling time using the numbers on a clock face. For example, 9 + 7 in normal arithmetic equals 16. However, in modular arithmetic with a modulus of 12 ("mod 12" arithmetic, also called *clock arithmetic*), the result of 9 + 7 is 4. If it is 9 o'clock in the morning then seven hours later it will be 4 o'clock in the afternoon. Because the hour number starts over after it reaches 12, the modulus is 12. In normal arithmetic, the result of adding two numbers grows as the numbers being added are larger. With modular arithmetic, the numbers can grow just to the value of the modulus. Although this key exchange system was a great breakthrough in cryptography, it still required that Alice and Bob exchange several messages to establish the shared secret key. The Diffie-Hellman key exchange protocol was also fundamentally a two-party protocol rather than a broadcast protocol that allowed Alice or Bob to communicate securely with



B.12.6. Whitfield Diffie and Martin Hellman are the inventors of the Diffie-Hellman key exchange protocol. This is remarkable process by which Alice and Bob can agree on a secret key using an open link that is vulnerable to access by an eavesdropper, Eve.

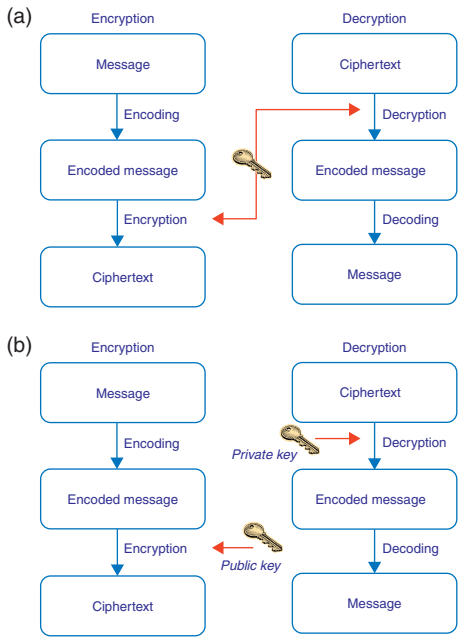


Fig. 12.14. (a) Symmetric encryption shares the same encryption key between sender and receiver. (b) Asymmetric encryption uses a different encryption key at each end of the communication.

others. Another breakthrough was needed to arrive at a secure and convenient cryptographic method that eliminated key exchange bottlenecks.

Up until 1975, all the encryption techniques in history had been *symmetric*, meaning that the key to unscramble the message was the same as the key used to scramble it in the first place (Fig. 12.14a). In the summer of 1975, Diffie outlined the idea for a new type of cipher that used an *asymmetric key* pair, one in which the encryption key and the decryption key were different but mathematically related (Fig. 12.14b). Although he showed that such a system could work in theory, Diffie was unable to find a suitable one-way function to actually carry out his idea. If such a system could be found, then it could work as follows. Alice would have two keys, one for encryption and one for decryption. She can make her encryption key public, her "public key," so that everyone has access to it, but she keeps her decryption key secret as her "private key." Now if Bob wants to send a message to Alice, he can encrypt his message using Alice's public key. When she receives the message, Alice is able to decrypt the message using her private key, secure in the knowledge that Eve, who only knows Alice's public key, would be unable to make sense of the message. This is the essence of the cryptographic system called *public-key cryptography*.

RSA encryption and pretty good privacy

The race to make asymmetric ciphers a reality was won by three researchers working in the Laboratory for Computer Science at MIT: Ron Rivest, Adi Shamir, and Len Adleman (B.12.7). Their resulting scheme is now known as *RSA encryption*, and it depends on modular exponentiation and the difficulty of factoring large numbers. The scheme relies on the fact that multiplication of two large prime numbers, p and q, to get the number N is very easy and

RSA-129

3490529610		3276813299		114381625757888867669
8476509491		3266709549		923577997614661201021
4784961990		9619881908		182967212423625825619
3898133417	X	3446141317	=	842935706935245733897
7646384933		7642987992		783059712356395870501
8784399082		9425397982		589890751475992900261
0577		88533		879543541

Fig. 12.15. The number RSA-129. Multiplication is computationally "very easy" whereas factorization of a number into its constituent prime numbers is computationally "very hard." This is the basis for the security of the RSA Public-Key Cryptographic system.

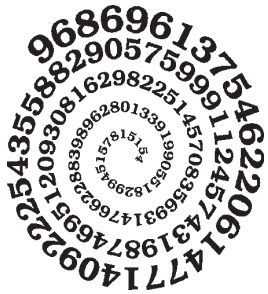


Fig. 12.16. The original encrypted text of Gardner's challenge.

quick to do with a computer, but the reverse problem of factoring N – deducing the prime numbers that when multiplied together produce N – is very difficult. Martin Gardner, in his "Mathematical Games" column in *Scientific American*, explained public-key cryptography and the RSA asymmetric cipher in August 1977. He issued a challenge to his readers by giving them a ciphertext to decode that had been encrypted using a public key N, which he published. The public key was a 129-digit number known as RSA-129. To decrypt the message, the readers had to *factor* (break up) this number into its two *prime factors*, the prime numbers that were multiplied together to produce the 129-digit number. It was almost seventeen years before a team of six hundred volunteers assembled sufficient computing power to discover the two prime number factors (see Fig. 12.15). When at last deciphered, Gardner's message read, "The magic words are squeamish ossifrage" (Fig. 12.16). Nowadays, given the huge increase in computing power since 1977 generated by Moore's law, much larger values than RSA-129 need to be used to secure messages and information. These numbers are so large that it is estimated it would take all the computing resources on the planet many thousands of years to factorize such large numbers. However, as we will see later, such public-key systems could be vulnerable to attack by a quantum computer if such a computer could be built.

In the 1980s, only governments, the military, and large businesses had computers that were powerful enough to use RSA efficiently. Phil Zimmermann (B.12.8), a software engineer specializing in cryptography and data security, believed that everyone should have the same guarantee of privacy in communications made possible by RSA encryption. Such a capability is particularly important for human rights activists operating in countries with repressive regimes. Even in more open countries, privacy of communications can be regarded as a basic democratic freedom. The problem is that this same freedom would also severely limit the ability of governments to monitor communications between criminals.

Zimmermann wrote a program that he called *Pretty Good Privacy* (PGP), a name inspired by *Ralph's Pretty Good Grocery*, a business in Garrison Keillor's fictional town of Lake Wobegon. In his PGP program, Zimmermann implemented a fast version of the RSA public-key system. Unfortunately, he also chose to ignore the fact that the RSA technology was patented. Zimmerman apparently

B.12.7. Ron Rivest, Adi Shamir, and Len Adleman are the inventors of RSA public-key cryptography protocol, which is now in widespread use. In their scheme, Alice now has two keys – an encryption key that she makes public and her private decryption key.

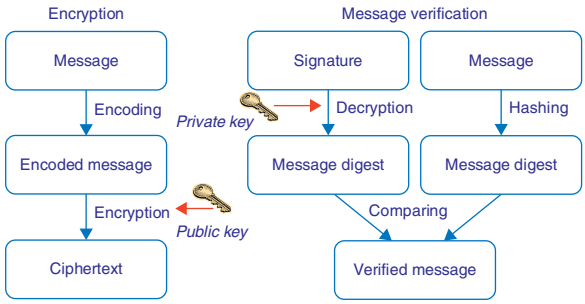


Fig. 12.17. The mechanism of a digital signature, an electronic signature that can be used to authenticate the identity of the sender of a message or the signer of a document.

hoped that the patent owner, Public Key Partners, would give him a free license since PGP was intended for use by individuals and not for commercial use. It was left to a group of cryptography researchers at MIT to make PGP legal by removing Zimmerman's implementation of the RSA algorithm and replacing it with a legal version with an appropriate RSA license.

The PGP software also incorporated *digital signature* authentication. Digital signature technology addresses the problem that, without a handwritten signature, it is difficult to be sure who actually sent an email message. Bob can use Alice's public key to send an encrypted message to her, but so can Eve, masquerading as Bob. So how can Alice check that the message is really from Bob? One way of verifying that the message was indeed sent by Bob goes as follows. Bob first encrypts the message using his private key and then does a second encryption, encrypting the resulting message using Alice's public key. When Alice receives the message, she begins by decrypting it by first using her private key and then uses Bob's public key to decrypt the still encrypted message. This way she can verify that the message came from Bob (Fig. 12.17).

In 1991, Zimmermann became worried that the U.S. Senate would pass a bill that would outlaw the use of such encryption technology, so he arranged for his PGP code to be posted on an Internet bulletin board. In response to this, the U.S. government, concerned about its ability to decipher communications between criminals or terrorists, accused Zimmermann of illegally exporting weapons technology. After some difficult years for Zimmermann, the government eventually dropped the case. Meanwhile, the code for the legal version of PGP was published in a book from MIT Press and could be legally exported from the United States. Ron Rivest summarized the basic argument against prosecuting Zimmerman as follows:

> It is poor policy to clamp down indiscriminately on a technology just 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. 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 wire-tapping, and the latter can thwart FBI fingerprint analysis. Cryptography and gloves

B.12.8. Phil Zimmermann is the creator of PGP, an email encryption software package. Originally designed as a human rights tool, PGP was published for free on the Internet in 1991. This made Zimmermann the target of a three-year criminal investigation by the U.S. government, which held that export restrictions for cryptographic software were violated when PGP spread worldwide.

> 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.[14]

Twenty years after the publication of Zimmermann's PGP software, strong encryption technology is now widely available, and governments and police forces round the world have had to adapt to the new reality.

Although encryption using PGP software provides a very high level of security, it proved too complex for the average Web user. Netscape introduced a procedure called the *secure sockets layer* (SSL) to protect e-commerce transactions over the Internet. Without intervention from the user, the browser and the web server use the SSL protocol to automatically exchange public keys and to agree on a third, secret *session key* to encrypt the information being transmitted only for the current session. Instead of using the http protocol, the link to the website now uses https (standing for HyperText Transfer Protocol Secure), which just applies the http protocol on top of a protocol called the Transport Layer Security (TLS) protocol, the successor to the SSL protocol. All the user sees is a padlock icon in the browser window. Clicking on the padlock gives the user a security report, which says, "This connection to the server is encrypted." The report also gives details of a *digital certificate*, a credential that certifies the identity of the remote computer. The certificate verifies that the public key belongs to the specific organization or owner of the website. An organization called a *certificate authority* (CA) issues digital certificates. The CA is what is called a "trusted third party" – that is, an organization trusted by both the subject of the certificate and by the user wishing to access that site. The result of all these measures is that users now have a secure channel by which they can communicate personal details such as credit card numbers or their Social Security number.

Cookies, spyware, and privacy

Web cookies were first used in communications over the Internet by Lou Montulli, a programmer at Netscape Communications in 1994. The company was developing e-commerce applications and wanted to find a way to keep a memory of a user's transactions so that it would be easy to implement a virtual shopping cart. A web cookie, also known as an *http cookie*, is a small amount of data that is sent from the website a user is visiting and stored in the browser on the user's computer. They were designed to provide a way for websites to remember the user's browsing activity. Cookies were first introduced into Netscape's browser in 1994 and into Microsoft Internet Explorer in 1995. Although the cookies were stored on the user's computer, users were not initially notified of their presence. Cookies are convenient in that they can be used to store passwords and credit card details. When a user revisits a website, the website can recognize the user through the information stored in the cookie.

The real threat to privacy, however, came with the introduction of *third-party tracking cookies* (Fig. 12.18). First-party cookies are associated with the IP address shown in the address bar of the user's browser. Third-party cookies are cookies that are downloaded from a different domain than that shown in the browser. These come about as follows. When a user downloads a web

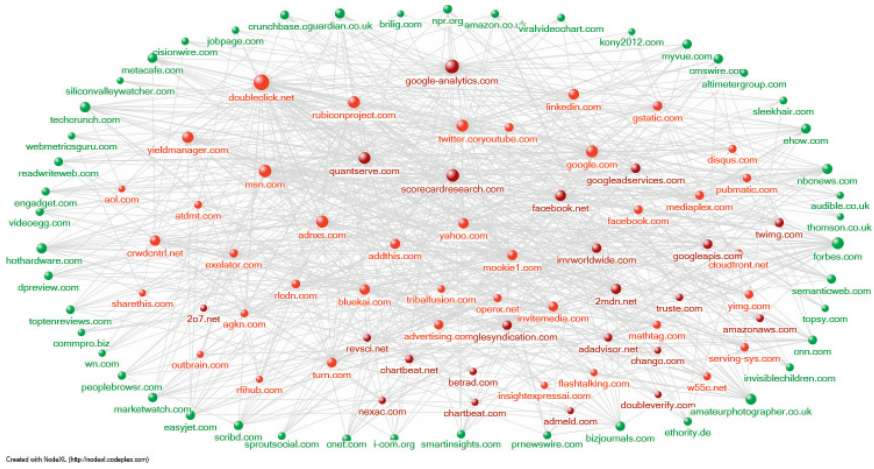


Fig. 12.18. Third-party cookies allow tracking companies and ad brokers to track the browsing behavior of web users. The green circles are the websites visited by a user and the purple circles are the companies analyzing the user's behavior and selling the information to the red sites that serve targeted advertisements to the user.

page this may contain an advertisement linking back to a different website. This site sets a cookie that tells the ad broker service that the user clicked on this web page. When the user visits another website the same thing happens and another cookie is downloaded. In this way, an ad broker can build up a complete picture of the user's browsing history. This information can then be sold to advertising agencies that can generate targeted, personalized ads, specific to the interests of the user, as revealed by their browsing history.

Spyware is software that can hide itself on a computer and gather and transmit information back to a black hat without the owner's knowledge. Spyware is different from viruses or worms in that the software does not try to replicate itself or spread to other computers. The Trojan horse software used by the Berkeley hacker is a form of spyware, for collecting user login information. Spyware can also collect other types of data such as bank and credit card information. In addition, spyware can track the user's Internet activity and serve annoying pop-up ads or change the computer's security settings and disable antivirus software. Cookies are a form of spyware and antispyware software now usually reports the presence of third-party cookies and offers ways to remove them.

Cookies have serious implications for the privacy of Internet users. In 2000, the U.S. government established strict rules for setting cookies, and modern browsers now offer users the option to block all cookies. In its *Directive on Privacy and Electronic Communications* in 2002, the European Union introduced a policy requiring a user's consent for setting cookies. It stipulated that storing data on a user's computer can only be done if the user is provided with information about how this data will be used. This was later relaxed to exempt first-party cookies – as in virtual shopping carts – from this requirement of obtaining prior user consent.

Key concepts

- Buffer overflow
- Trojans, viruses, and worms

- Rootkits and botnets
- Cyberespionage and cyberwarfare
- Cryptography and key exchange
- One-way functions and RSA encryption
- Cookies and spyware

Etymology of "spam"

The name *spam* derives from a famous 1970 Monty Python sketch about a man and his wife ordering food in a café, where they are offered various menu items all based on SPAM, a trade-named canned meat product:

Man: Well, what've you got?
Waitress: Well, there's egg and bacon; egg sausage and bacon; egg and spam; egg bacon and spam; egg bacon sausage and spam; spam bacon sausage and spam; spam egg spam spam bacon and spam; spam sausage spam spam bacon spam tomato and spam …
Vikings [starting to chant]: Spam spam spam spam …
Waitress: … spam spam spam egg and spam; spam spam spam spam spam spam baked beans spam spam spam …
Vikings [singing]: Spam! Lovely spam! Lovely spam!
Waitress: … or Lobster Thermidor a Crevette with a mornay sauce served in a Provençale manner with shallots and aubergines garnished with truffle paté, brandy and with a fried egg on top and spam.
Wife: Have you got anything without spam?
Waitress: Well, there's spam egg sausage and spam, that's not got much spam in it.
Wife: I don't want ANY spam![15]

God rewards fools

As Whitfield Diffie and Martin Hellman pursued the key distribution problem, they were joined by graduate student Ralph Merkle, who shared their enthusiasm for solving what seemed to be an impossible problem. Hellman commented:

> Ralph, like us, was willing to be a fool. And the way to get to the top of the heap in terms of developing original research is to be a fool, because only fools keep trying. You have idea number 1, you get excited, and it flops. Then you have idea number 2, you get excited, and it flops. Then you have idea number 99, you get excited, and it flops. Only a fool would be excited by the 100th idea, but it might take 100 ideas before one really pays off. Unless you're foolish enough to be continually excited, you won't have the motivation, you won't have the energy to carry it through. God rewards fools.[16]

A truly cryptic development

An interesting postscript to the encryption story takes us back to the secrecy that surrounded the cryptographic work on the Enigma and Lorenz codes at Bletchley Park during World War II. After the war, the government of the United Kingdom concentrated its code-breaking efforts in a new agency called the Government Communications Headquarters (GCHQ) in Cheltenham (Fig. 12.19). By the 1960s, the British military had recognized the need for secure communications between troops in the field but was concerned about the logistics and costs of key distribution. GCHQ set some of its researchers on the problem, and the result was that, as Simon Singh says: "By 1975, James Ellis, Clifford Cocks and Malcolm Williamson had discovered all the fundamental aspects of public-key cryptography, yet they all had to remain silent."[17] It was not until 1997 that Cocks was finally allowed to present a brief history of GCHQ's independent discovery of public-key cryptography. The same zeal for secrecy of successive U.K. governments denied Tommy Flowers meaningful recognition for his pioneering work in building the Colossus computers after the end of World War II.

Fig. 12.19. An aerial photo shows the GCHQ, the British agency responsible for communications security, based in Cheltenham, U.K. Two Colossus computers from Bletchley Park went to GCHQ after the war; the remaining eight were destroyed on Churchill's orders.