Trustworthiness: Reliability and Security

Networked computing is acceptable in mission-critical applications only if it is trustworthy. This means the application works correctly, almost all the time, and is secure against various external threats and natural disasters. As the applications of networked computing expand to encompass critical infrastructure, business applications, and electronic commerce, trustworthiness becomes a crucial issue for business and the larger society. The impact of failures or inappropriate penetration could be widespread and damaging.

8.1 Facets of Trustworthiness

Several major facets of trustworthiness are listed in Table 8.1. Any trustworthy application must combine these elements. What must be done right to have hope for a trustworthy system? The answer is many things, including important human and management elements. Further, a computer system is only as trustworthy as its weakest link. These elements must not only be combined carefully, but they also interact.

8.1.1 Program and System Correctness

A major source of unreliability is flawed computer programs. Software applications and the infrastructure that support them are very large, complex, and can experience an astronomically large number of possible conditions (see Section 6.1.1 on page 178). Although any application is subjected to extensive testing, including alpha- and beta-site testing with actual users (see Section 3.4.2 on page

Table 8.1 Important aspects of networked computing system trustworthiness.

Factor	*Description*	*Analogy*
Intrinsic reliability	The application operates correctly, all the time, even in the face of inevitable equipment and facility failures and natural disasters. This assumes no deliberate internal or external threats or vandalism.	A bank's financial systems should be designed and managed such that, in the course of normal operations, customers are served well and no financial errors are made. This includes contingency plans for major disasters such as floods and earthquakes.
Security	The system continues to operate reliably and protects its confidential information in the face of aberrant or malicious behavior on the part of unauthorized parties.	The bank designs its facilities and operations to prevent criminal behavior such as robberies, sabotage, and theft of confidential information.
Human elements	Trustworthiness is dependent on competent and honest operators, who faithfully follow established policies and procedures and make good decisions. A crucial component of security is operational vigilance for unusual or suspicious activity.	No matter how well the bank's systems are designed, they are vulnerable to problems caused by incompetent or dishonest workers. The bank posts guards at branches and crucial facilities to report suspicious activity and deter lawbreakers.

106), it is simply not possible to thoroughly test all possible conditions. Thus, latent program errors (colloquially called software bugs) are inevitable.

ANALOGY: *The transportation system works well most of the time, but occasionally there are accidents.*

These bugs are observed, and corrected as they are observed, as part of maintenance and new releases. Serious flaws may be manifested at the time of installation, in which case it may be necessary to "roll back" to a previous release. Serious bugs can cause the application or computer system to *crash*, meaning an unrecoverable error necessitates a restart from scratch.

EXAMPLE: *The Nasdaq stock exchange upgraded the software on both its primary and backup computers (for compatibility) in July 1994. This caused their systems to crash, and the exchange was forced to halt trading for two hours while the previous release was restored [OST97].*

Bugs in their most benign (but most difficult to detect) form don't cause crashes but "merely" incorrect results.

EXAMPLE: *The Pentium processor was found to encounter errors in its floating point arithmetic, caused by a bug in its* microcode *(software embedded within the chip). Although it rarely occurred, when the bug was detected, Intel was forced to supply replacements to many customers.*

Human Element

Every application has to be configured during installation for its local conditions, and during its operations innumerable decisions are made by its operators. Even the best laid plans of its designers are dependent on vigilance and competent decisions during installation and operation.

EXAMPLE: *A telephone switching center in New York City inadvertently went on battery backup (during a power system overload compounded by a backup generator failure) in September 1991. The designers had included alarm bells and warning lights to warn of this condition, but they were ignored by operators for six hours until the batteries discharged. Because the center was crucial to local air traffic control, 400 flights were canceled and tens of thousands of passengers were inconvenienced [OST97].*

Emergent Behavior

A large application is decomposed into individual modules whose behaviors are individually well understood. However, when many modules interact in complex ways (see Section 6.1.1 on page 178), surprising and unanticipated behaviors (called *emergent behaviors*) are sometimes observed. Emergent behavior is not an error per se; rather, it is a natural phenomenon that can be beneficial but is most often undesirable or even destructive.

Diversity, Reliability, and Security

A diversity of technologies and vendors may increase the likelihood of either downtime or security holes, in part because the operators have less familiarity with each individual technology. On the other hand, in the absence of such diversity, problems that do occur can be more severe.

ANALOGY: *Genetically diverse animal populations are more resistant to disease. If all animals in a population were genetically identical, they could readily succumb to the same disease.*

When an application incorporates multiple identical copies of the same program, very minor errors can spawn serious problems because of subtle interactions among the multiple copies. In this regard, software components (see "What Is a Software Component?" on page 200) are a double-edged sword. On the one hand, widely used components are more thoroughly tested and thus have fewer bugs. On the other hand, since they are distributed more broadly, they might interact in surprisingly destructive ways.

EXAMPLE: *A small flaw in a software upgrade in a telephone switch in New*

ANALOGY: *A system as complex as the worldwide economy exhibits emergent behavior, such as business cycles and recessions, that cannot be well anticipated or explained in terms of the behavior of individual economic actors. On the other hand, by studying the behavior of the system in the large (in macroeconomics), insights—if not complete understanding—can be gained.*

Emergent behavior can cause crashes, but more benign forms cause deterioration in performance or functionality. Prevention of emergent behavior, or even of recurrence, is difficult. The widespread replication of the same technology seems to increase the likelihood of emergent behavior (see the sidebar "Diversity, Reliability, and Security").

Availability

Availability refers to the fraction of the time an application is running when it is needed. Availability is reduced by *downtime*, which is a period of unavailability because of scheduled maintenance (including installation of software upgrades, or fixing of bugs or configuration errors or problems), hardware failures, recovery from crashes, power failures, etc. Availability is usually expressed as the percentage of time the system is available. The relationship of downtime and availability is shown in Table 8.2.

There is a trade-off between the economic impact of downtime in comparison to the cost of high availability. In many essential systems, such as an air traffic control system or telephone network, the social and economic cost of downtime is so great that extraordinary measures (and costs) are undertaken to ensure high availability (1 hour per 20 years is achieved in telephone switching systems). In many other applications, lower availability may be tolerated in the interest of lower cost. For example, 99 percent availability might be acceptable in a typical business application, if the downtime came in many short periods rather than one long failure per week, or if it could be scheduled at less critical times (2 a.m. Sunday morning, for example) [Ber97].

Downtime is not the only impact of application crashes or aberrant behavior. There may be permanent loss of data, which has its own

Table 8.2 Relationship of downtime to availability [Ber97].

Downtime	*Availability*
1 hour per day	95.8%
1 hour per week	99.41%
1 hour per month	99.86%
1 hour per year	99.9886%
1 hour per 20 years	99.99942%

economic or social cost. The issues are thus how often downtime occurs, what is the restoration time, and how much if any data is lost.

High availability requirements can increase the costs of development, equipment, and worker training or salaries (hiring more skilled operators). Development costs are increased by greater attention to recovering gracefully from unusual conditions and more time and attention paid to program correctness and testing. Avoiding downtime even during inevitable failures of hardware and communication links requires *fault tolerance*, in which a failed piece of equipment is automatically replaced by a redundant replacement. Some useful technical tools for increasing availability are listed in Table 8.3.

These features are complementary. For example, if one processor fails, switching to a redundant processor will prevent loss of data if there is a replica, or if the data is persistent. Data replication has many uses besides fault tolerance (see the sidebar "Uses of Data Replication").

8.1.2 Security: Countering External Threats

Intrinsic reliability addresses unintended or natural problems. Security addresses hostile threats from workers or citizens with nefarious purposes, such as vandalism (corrupting data or disrupting operations) or theft of data or services. The greatest external threats arise

York City in January 1990 cascaded throughout the country, causing over 50 percent of the telephone traffic to be blocked in the AT&T nationwide network for over seven hours [OST97]. The same upgraded software had been replicated across the country, and the replicated flaw interacted, causing each switch to crash the switches with which it interacted.

Similarly, if a security flaw is exploited in a particular technology, the damage is potentially more severe if that technology is widely deployed.

Uses of Data Replication

There are a number of motivations for data replication besides aiding in fault tolerance. Replication can improve performance by keeping a replica of data closer to where it is needed. In networked computing, usually "closer" means "on this side of a communication link bottleneck," which leads to the idea of *caching* described in Chapter 12.

ANALOGY: *If you are living in a two-story house and don't appreciate the time and effort to run up and down the stairs, you might purchase two copies of a book you are reading and keep one copy on each floor. That way, a copy will be close at hand at all times.*

Similar issues arise when users collaborate on common information, as in collaborative authoring (see the sidebar "Collaborative Authoring" on page 28). The data must be replicated, one copy for each user. Replication of data where the copies are all being modified simultaneously means that changes to any one copy must be reflected as soon as possible to the other copies. A similar issue is document *version control*. When multiple users are making changes to a document, there is a need to recover older versions of the document if not all users agree with the changes.

Table 8.3 Techniques for fault tolerance and graceful crash recovery.

Feature	*Description*	*Analogy*
Equipment redundancy	Provide backup equipment (hosts, network switches, communication links, etc.). Upon failure of the primary equipment, switch to the redundant backup.	Keep a spare car in the garage. If you have car trouble, get a ride home to use your spare car.
Data replication	Keep replicated copies of data. Upon loss of data, restore from the replica (losing changes in the interim).	Leave a spare copy of your income tax returns with a friend. In case a fire destroys either copy, use the spare.
Data persistence	A property of data that outlives the program that created it (see Section 6.3.1 on page 207). Upon a crash, data can be recovered when the program is restarted.	Take written notes at a meeting. In case you can't remember what happened, fall back on the notes.

when hosts are connected to a network, especially the global Internet, since this allows virtually anybody to attempt access. Various means have been developed to counter these threats, although not yet with complete effectiveness.

ANALOGY: *Once access to bank branches is given to the general public, both customers and bank robbers can enter. Countermeasures such as glass partitions, surveillance cameras, and alarms can deter criminals but also create a less friendly and functional environment for the customer.*

This analogy illustrates two downsides to security. First, it is expensive—like high availability, high security increases design costs, capital investments, and operational costs. Second, security is invasive—there is a trade-off between helping legitimate users accomplish their purposes easily and preventing vandalism or theft.

Good security requires an understanding of both the inherent vulnerabilities of the application and the nature of potential threats. Unfortunately, such comprehensive understanding is infeasible, so vigilance and continual learning of new threats are necessary

[OST97]. Security is an ongoing process, including new measures to counter newly identified threats [CST98].

EXAMPLE: *CERT is a clearinghouse located at Carnegie Mellon University that "studies Internet security vulnerabilities, provides incident response services to sites that have been the victims of attack, publishes a variety of security alerts, researches security and survivability in wide-area-networked computing, and develops information to help you improve security at your site."*

Two aspects of security are easy to overlook, so they should be emphasized up front:

- An individual security measure is rarely effective in isolation. Security is by nature a *system* composed of many complementary measures, and they are only effective in concert.
- People are as important to security as technology. Although people external to an organization or administrative domain are a security threat, employees or other insiders are a much greater threat, as they have greater opportunity for mischief. Conversely, worker training, competence, and vigilance (fortunately more the norm than dishonesty and maliciousness) are the most important ingredients of effective security.

Analogy: The Postal System

It is useful to consider briefly an analogy to the postal system, which encounters in the physical world many of the same security objectives and issues as networked computers. Some features of postal system security are illustrated in Figure 8.1, focusing specifically on sending a document through the mail.

Assume that Alice wants to send a message (using a postal letter) to Bob. To begin, Alice writes her message on paper using permanent ink, making it difficult to modify surreptitiously. This provides assurance to Bob that the message was not modified in transit by some third party, say Eve. (The fictitious names Alice, Bob, and Eve are commonly used by computer security people to keep straight the parties to a security protocol.) Formally, the paper and permanent ink provide evidence of the *integrity* of the message. Also, to assure

Availability, Security, and the Market

High availability computer systems were pioneered by Tandem Computer in the 1970s, and since then other vendors have entered the market. These systems are used in applications such as OLTP (see Section 2.6.1 on page 53), where downtime is expensive.

The market is arguably less effective in fostering security. Threats are diffuse and difficult to characterize, and the cost benefits of security are difficult to quantify. Much core technology (the UNIX operating system and the internet technologies) were developed in a benign environment of trusted users (the university research community) and thus were originally lax on security.

The most effective market mechanism for promulgating good security is to make sure the risks (and costs) of security breakdowns are borne by those in a position to deploy the security necessary to counter those risks.

EXAMPLE: *In the United Kingdom, by law, the risk of theft of money from automatic teller machines (ATMs) is borne by the consumer, whereas in the United States it is borne by the bank. As a result, banks in the United States have deployed much stronger ATM security measures.*

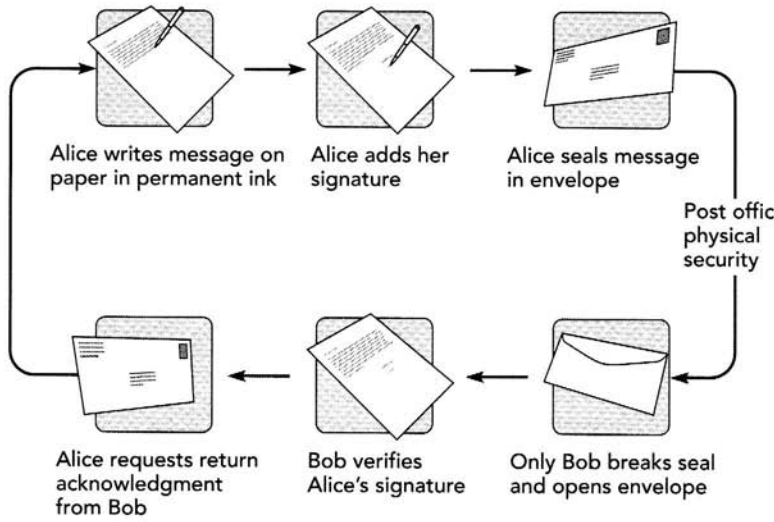

Figure 8.1 How the postal system provides some security features in a message from Alice to Bob.

Bob that Alice was the source of the message (and not Eve pretending to be Alice), Alice may affix a signature that Bob can check against a copy of her signature that he keeps in his files. The signature provides *authentication*—a provable identification of Alice. Alice may also not want Eve to read the message, which she would like to keep confidential (between her and Bob). For this purpose, Alice seals her message in an envelope to keep away prying eyes. During transit, the postal system maintains physical security, allowing access only to postal workers, who are presumed honest and trustworthy. Once it arrives, Bob can unseal the envelope to read the message and examine the signature to authenticate Alice. He may notify Alice (using a return letter) of successful delivery. Later, Bob may find it necessary to prove in court that Alice sent him the message (for example, if it is a legal contract). For this purpose he would produce the message and argue that since it was written in permanent ink on paper—and thus difficult to modify without detection—and contained Alice's signature, Alice must have sent it. The inability of Alice to later deny she sent the message is called *nonrepudiation*.

It is important to note that some of these security measures would falter if they had not been combined. For example, Alice might be able to repudiate the message if she had not been authenticated (by her signature) or if she could claim the message was altered in transit (the message didn't have integrity).

Even with all these measures, the postal system has inherent vulnerabilities. Bob's mailbox—in order to be accessible to a postal worker—is also accessible to any passerby (such as Eve), who can steal a letter or insert a fake letter. The envelope also allows Eve to determine that Alice sent Bob a message (by her return address), even if she can't examine the contents. To counter some of these threats, the government has established *legal sanctions*—including laws against the theft of mail from, or insertion of unauthorized mail into, a mailbox. If Bob is especially worried about these threats, he could rent a post office box, which restricts access to his mail (by giving him an exclusive key) and thus maintains physical security all the way from Alice to him.

Threats to Networked Computing

Many of the security measures in the physical world must be transferred to networked computers. Messages crossing the network, whether they originate with a user (such as an email message) or with an application (such as module-to-module interaction) are data (a collection of bits). This is a security challenge, because the infrastructure, in order to manipulate and process those bits (for example, replicate them—see Section 5.4.1 on page 157), must inherently be able to observe and modify them. Without additional measures, there would be no way to prevent a message from being examined or modified in transit, or to detect that this happened.

Similar observations apply to the other security elements identified in the postal system analogy. Each message must include identification of sender and recipient, also represented by bits, which can easily be observed or modified. In an internet, for example, it is possible to pretend to be a different host by faking the sender identification information (this is called *spoofing*).

Computer Viruses

Between executions, programs are stored in a special *executable* file in the computer storage (such as magnetic disk). During these dormant times, programs are vulnerable to infection by a virus. A virus attaches itself to an executable (called a host) in a way that causes its own execution each time the host is invoked. The virus can then replicate itself by attaching its own replica to other executable files, and at the same time, it can consume resources or destroy data.

ANALOGY: *In biology, a virus is an infectious agent that can multiply by replication only in living cells of animals, plants, or bacteria. Viruses can consume resources or act destructively. Similarly, computer viruses infect computer systems and can multiply by self-replication.*

Viruses must be taken seriously whenever an executable is moved from one host to another, using a floppy disk or transferring over the network. Fortunately, there are excellent utilities for detecting and eradicating viruses.

ANALOGY: *It is easy to put a deceptive return address on a postal envelope. You are wise not to trust the return address as a form of authentication, but to look for a more secure authentication of the sender (such as a signature) inside the envelope.*

Networked computers are susceptible to more subtle threats as well. One threat, particularly in a network environment, is the *computer virus.* This is a piece of executable program that can "infect" computer files, self-replicate, and cause damage (see the sidebar "Computer Viruses").

Like criminals that might break into a post office, hosts are susceptible to *intrusion* by unauthorized access through the network. Ways are needed to prevent these intrusions by admitting authorized users and preventing access to others. Frequently it is desired to provide *conditional* access to some applications but not others.

EXAMPLE: *A Web server is often intended to be accessed by anybody on the Internet. At the same time, there may be other applications or stored information on the same host that should have restricted access.*

Even if access is denied to any but authorized users, others might gain entry by spoofing the identity of an authorized user. To counter this attack, there have to be user authentication measures analogous to the signature in the postal system. Similarly, if legal contracts are "signed" over the network, or orders for goods placed, a forgetful or unscrupulous user might later repudiate the transaction unless there are security measures to prevent this.

A malicious attacker may mount a *denial of service attack*, injecting vast amounts of artificial work or communications causing a host or network to become overloaded and degrading the performance for legitimate users.

ANALOGY: *Your teenage child might retaliate for the punishment of being banished to his room by tying up the phone.*

EXAMPLE: *In 1988, a student at Cornell University unleashed a* worm *on the Internet that resulted in a denial of service to a large num-*

ber of UNIX hosts. A worm is similar to a virus (see the sidebar "Computer Viruses" on page 248), except it replicates by exploiting operating system vulnerabilities to get itself installed and executed on another computer.

8.2 Computer and Network Security Measures

How can the myriad external threats to networked computer systems be countered? This is the role of security measures. Networked computing security includes a number of building blocks analogous to those in the postal system, which are listed in Table 8.4. Fortunately, these capabilities are sophisticated and capable of achieving a much higher level of security than in the postal system analogy. However, to be effective, they have to be *used*, which unfortunately can be invasive (in cost or convenience) to legitimate users. Among other things, these security measures create serious administrative or operational issues.

The most important point is that the capabilities in Table 8.4 work in concert and must be deployed in a coherent *system* backed by well-trained and vigilant operators, a coherent set of established security policies, and criminal laws and penalties. Again, the overall security is only as good as the weakest link.

8.2.1 Encryption Ensures Confidentiality

The confidentiality of data stored or communicated across the network is achieved by *encryption*. Suppose Alice wants to send a confidential message to Bob while preventing an eavesdropper, Eve, from reading the message. The idea is for Alice to *lock* (encrypt) a message so that it can only be *unlocked* (decrypted) by Bob, who has the unlocking *key*. Eve is unsuccessful in unlocking the message to view it because she does not possess Bob's unlocking key.

ANALOGY: *In the physical world, a* lockbox *can be used to provide confidentiality. The lockbox requires an unlocking key to open. To be analogous to encryption, a special lockbox that requires two keys—a locking key to lock and an unlocking key to open—must*

Table 8.4 Pillars of a computer security system.

Capability	*Description*	*Analogy*
Authentication	Messages received over the network may be received from anyone and anywhere. Authentication enables a recipient to verify the identity of a sender.	On paper documents, a signature is appended. The recipient of a telephone call may be able to identify the caller by recognizing her voice.
Message integrity	A message is a collection of bits and may easily be modified as it crosses the network. Message integrity ensures that the message has not been modified since created by the sender.	Paper documents created in permanent ink are difficult to modify without detection. Although additions are possible, they are difficult to achieve without detection.
Confidentiality	A message passing the network might be read by an eavesdropper. Confidentiality ensures that a message can only be read by the intended recipient.	The sender of a letter encloses it in an envelope to keep away prying eyes. A postcard, on the other hand, does not provide confidentiality.
Nonrepudiation	A sender might later claim never to have sent a message. Nonrepudiation enables a recipient to prove (in court if necessary) that the sender did send the message.	A recipient of a document (purchase order or contract, for example) will require a signature on it so he can later prove the identity of the sender. Additional assurance is provided by signing two copies of the document, one kept by the sender and one sent to the recipient.
Access control	A host connected to the Internet will be vulnerable to theft or vandalism by unauthorized parties. Access control restricts access to hosts, or particular applications, to authorized and authenticated users.	Postal patrons are allowed into the public area of the post office, but only postal employees can enter secure areas where mail is sorted. Employees are authenticated by a picture badge.

be assumed. First, Alice locks a message in the lockbox to protect it from being read by Eve. Second, Bob unlocks the lockbox to obtain the contents—the message from Alice.

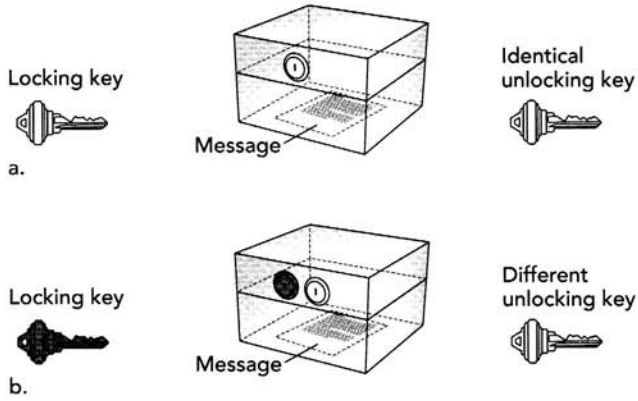

Figure 8.2 Two types of lockboxes, which are analogous to a symmetric and asymmetric encryption system.

Encryption comes in two distinct flavors: symmetric and asymmetric. These are analogous to two distinct types of lockboxes, depending on the relationship of the locking and unlocking key, as illustrated in Figure 8.2.

- *Symmetric locking and unlocking keys:* The lockbox requires identical keys to lock and unlock the lockbox. There are two replicas of this key—one held by Alice and one by Bob.
- *Asymmetric lock and key:* The lockbox has two different keys, a locking key to lock the lockbox (possessed by Alice) and an *unlocking* key (possessed by Bob) that unlocks the box. Alice's key cannot unlock the box—it can only lock it. Only Bob's key can unlock the lockbox to recover the message once it has been locked.

The symmetric lock and key is conceptually simpler, but there are advantages to the asymmetric lock and key, as will be seen shortly.

As shown in Figure 8.3, in networked computing a message is data (a string of bits) and the locking or unlocking key must also be data. The analogy to the lockbox is an *encryption algorithm*—a mathematical algorithm that takes as input the message and the key and generates as its output the encrypted message (also data, another

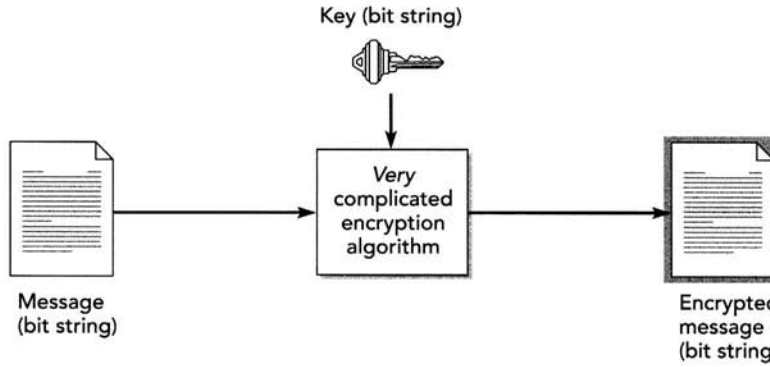

Figure 8.3 The encryption algorithm hides a message from anybody without a complementary decryption key.

string of bits) (see the sidebar "Any Information Can Be Represented by Bits" on page 10). This encryption algorithm is designed so that the original message can be recovered from the encrypted message by a *decryption algorithm*, which requires the unlocking key. Together, the encryption algorithm applied by the sender, followed by a decryption algorithm applied by the recipient, constitutes a *confidentiality protocol* (see Section 7.1 on page 216).

Strictly speaking, recovering the message from the encrypted message is not impossible. For example, Eve could try unlocking the message by systematically trying all possible unlocking keys. Rather, it must be assumed that this type of attack is *computationally impractical*. This means it is not possible—using a computer that Eve can afford—within a period of time that makes the resulting message useful to her.

The time it would take to try all keys depends on the *key length* (number of bits in the key) and the speed of available computers. The key length can be made sufficiently long to foil this attack for any desired period, for a particular computer technology.

EXAMPLE: *For a 128-bit key, there are $2^{128} = 3 \times 10^{38}$ different keys. If a computer could try 100 million keys per second (a rough time for today's fastest desktop computers), this attack would take 3×10^{30} seconds, or 10^{21} centuries.*

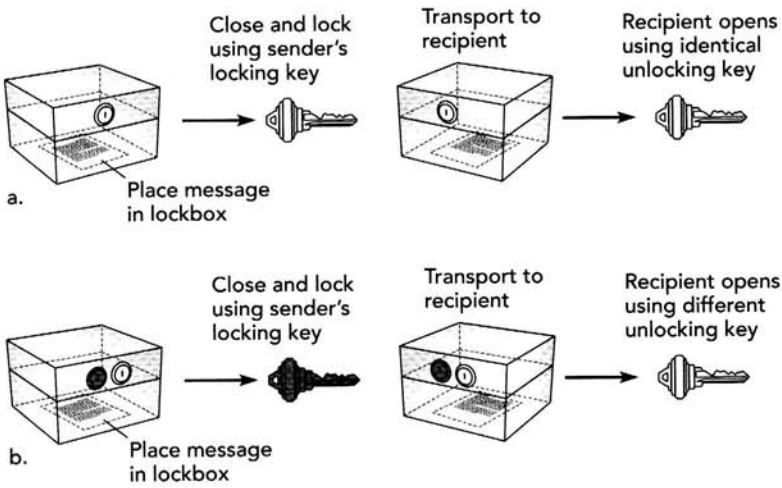

Figure 8.4 Using a lockbox to preserve confidentiality: (a) symmetric and (b) asymmetric.

Symmetric vs. Asymmetric Encryption

The lockbox analogy illustrates confidentiality for the two cases in Figure 8.4. The requirements for confidentiality are different in the two cases:

- *Symmetric encryption:* The locking and unlocking keys are identical. Eve must not possess a replica of this key.
- *Asymmetric encryption:* The locking and unlocking keys are different. Eve must not possess a replica of the *unlocking* key, but it doesn't matter if she possesses a replica of the locking key, because that would not allow her to unlock the message.

The last statement is important, but to be valid, an additional important assumption is necessary: It must be computationally impractical for Eve, possessing a replica of the locking key, to turn that into a replica of the unlocking key. Fortunately, there are asymmetric encryption algorithms with this property.

In the asymmetric case, it is permissible to distribute replicas of the locking key publicly. Bob, who wishes to receive confidential messages from many people (including Alice), can give a replica of the

locking key to anyone he expects might want to send him a confidential message. Bob need not worry that Eve will then possess a replica of the locking key, since it will not help her unlock a message sent by Alice (or anyone else). The locking key can be made available to anyone to send Bob confidential messages (for example, it can be included in directories or published on Bob's Web home page). For this reason, the asymmetric locking key is called a *public key,* and the asymmetric unlocking key is called a *secret key* (since only Bob can possess it).

For the symmetric case, the single key must be a *secret* key, not divulged to anybody other than the sender and recipient. This makes the symmetric approach logistically troublesome to Bob, since he must somehow get a replica of this secret locking key to Alice. Obviously, he can't just send the unlocking key to Alice over the network, lest Eve capture it in transit. Further, he must trust Alice not to give it to anyone else or use it herself to read confidential messages to Bob from others.

EXAMPLE: *If Bob is a merchant selling goods over the network, literally anybody is a potential consumer who might buy Bob's goods, including Alice. If Alice wants to use her credit card to make a purchase, she would like to avoid its interception by Eve, who might use it improperly to make her own purchases. The asymmetric encryption protocol is preferred, since Bob can simply publish his public key, and Alice can use it to encrypt her credit card number. Eve, who does not possess Bob's secret key, cannot view the credit card number.*

To summarize, the two approaches are illustrated in Figure 8.5. In subsequent figures, an encrypted message is denoted by a shaded box as in Figure 8.5.

EXAMPLE: *The most widely used de jure symmetric encryption standard is the* data encryption standard *(DES), formally called DEA-1 by ISO (see the sidebar "International Organization for Standards (ISO)" on page 135). A widely used de facto asymmetric encryption standard is RSA, named after its inventors Rivest, Shamir, and Adleman.*

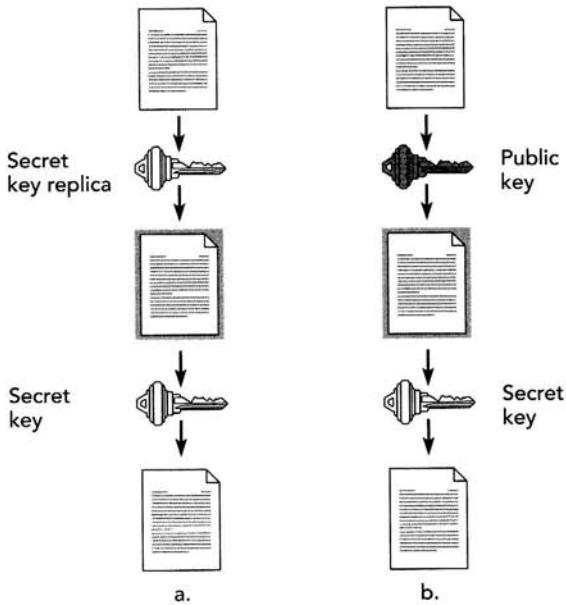

Figure 8.5 (a) Symmetric and (b) asymmetric encryption confidentiality protocols.

An important characteristic of the asymmetric encryption protocol is that the roles of the two keys can be reversed: Alice can encrypt a message using her secret key, and Bob can decrypt the result using Alice's public key, and the original message is recovered. This reversed protocol does not provide confidentiality, since anybody with Alice's public key can recover Alice's message. However, it will be useful later for authentication and message integrity.

This description might lead you to conclude that only asymmetric encryption is useful. Unfortunately, asymmetric encryption and decryption require roughly one thousand times more computational effort than symmetric encryption and decryption. Thus, symmetric encryption is superior for "bulk encryption" of large amounts of data and is widely used for this purpose. In practice, the two approaches are often combined (see Section 8.2.4 on page 262).

8.2.2 Authentication

Bob has to worry that he may be dealing with an impostor who *claims* to be Alice. Compared to the personal interaction of the physical world, impersonation is easy over the network.

EXAMPLE: *When Alice visits an Unlimited store in the local mall, there are a number of clues that this is a legitimate business—such as the store, the sign out front, etc. Over the network, Alice should be skeptical of a seller claiming to be Unlimited, since an unscrupulous person (such as Eve) could more easily set up a Web site claiming to be Unlimited and collect money from Alice without delivering goods. In their store, Unlimited authenticates Alice by requesting a picture identification or credit card. Over the network, Eve might more easily impersonate Alice, and Alice might then have to pay for goods actually ordered and received by Eve.*

Authentication allows Bob to verify that the person claiming to be Alice is in fact Alice. In practice, both Bob and Alice should authenticate one another. Examining how authentication is done in the physical world gives some clues as to how it can be done in networked computing:

- Bob could compare a person's face with Alice's picture shown in her picture identification card and, if it matches, authenticate her as Alice. This authentication depends on Alice's physical characteristics.
- If a person can produce Alice's credit card—and the issuer of the credit card confirms that Alice has not reported it stolen—Bob might authenticate that person as Alice. This authentication depends on the person presenting something only Alice should possess.

In both these cases, authentication depends on a trusted third party, called an *authority*—either the issuer of Alice's picture identification or the issuer of her credit card. In fact, it is impossible to authenticate Alice without the aid of an authority. Depending on someone to authenticate themselves is inherently suspect.

Biometrics

The two physical-world authentication techniques have analogs in networked computing. The first, based on a person's physical characteristics, is called *biometrics*.

EXAMPLE: *If Bob wants to authenticate Alice, and has obtained some unique physical characteristics of Alice from a trusted third party (her picture, fingerprint, or the pattern of the iris of her eye, for example), then Bob can verify that physical characteristic.*

Biometrics requires a trusted way to gather the biometrics data on the person being authenticated. This is most widely used to authenticate a person entering a physical facility such as a bank vault, computer center, or ATM, where the data is gathered under the complete control of the authenticator.

Shared Secret

Asking the person claiming to be Alice to produce something that only Alice should possess, as confirmed by an authority, is the most practical authentication over a network. In the physical world, that "something" is a physical object such as a picture identification or a credit card. Over the network, that something can't be a physical object and thus must be a *secret*. A secret is data—a string of bits—or something equivalent, such as a password, which only one person knows. A *password* is a nonsensical string of characters (letters, punctuation marks, numbers) that the person has chosen at random and committed to memory. Authentication proceeds by challenging a person claiming to be Alice to produce Alice's secret. There are at least several subtleties in doing this securely:

- It is a bad idea for Alice to give her secret to Bob for purposes of authentication, since Bob could later use it to impersonate Alice.
- It is doubly bad for Alice to send her secret to Bob over the network, as Eve could intercept it and later use it to impersonate Alice (and Bob would have it as well).
- There must be an authority that Bob trusts to verify Alice's secret.

There are thus three valued properties of an authentication scheme based on a secret:

- Bob should not be required to know Alice's secret in order to authenticate her.
- Alice should be able to prove she has a secret without revealing it.
- Bob should be able to verify the secret with an authority he trusts.

The first property can be achieved using asymmetric encryption keys. The idea is for Bob to acquire Alice's *public* key and then verify that the person claiming to be Alice possesses Alice's corresponding *secret* key. This is secure if Alice's public key is confirmed by an authority that Bob trusts. The second property can be simultaneously achieved using a *challenge/response protocol.*

EXAMPLE: *In a commonly used protocol, Bob generates a random integer k, encrypts it using Alice's public key, and sends the result to Alice with the challenge to respond with k + 1 encrypted by Alice's secret key. In order to meet that challenge, Alice has to decrypt the message to obtain k, add unity, and encrypt the result using her secret key. Both require Alice to have her secret key. Bob, knowing Alice's public key, can decrypt the response from Alice to confirm that the result is, in fact, k + 1. This protocol depends on the reversal of the role of the public and private keys in the asymmetric encryption protocol. Notice that it does not require Alice to reveal her secret key, or Bob to possess it.*

The third property is a bit more difficult. Bob must somehow verify Alice's public key with an authority he trusts. The way this works is similar to the driver's license authentication in the physical world. Alice presents Bob with a credential called a *digital certificate* (analogous to a driver's license) that was issued to her by a trusted authority (analogous to the motor vehicle bureau). The digital certificate provides Bob with Alice's public key in a way that he can trust.

Digital Certificates and Certificate Authorities

Bob must confirm Alice's public key with a trusted authority in order to authenticate Alice. A side benefit is that once Bob has Alice's

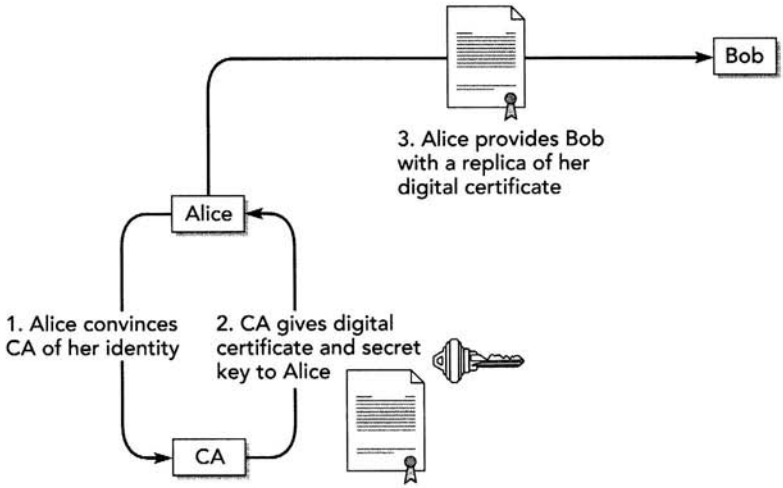

Figure 8.6 Alice uses a digital certificate to provide an authenticated public key to Bob.

public key, he can also use it to send Alice confidential messages—and be assured that Alice and only Alice can read them—as well as authenticate her. The trusted authority is called a *certificate authority* (CA), and what the CA provides in order to verify Alice's public key is called a digital certificate.

ANALOGY: *A digital certificate is analogous to a driver's license that Alice might use to prove her identity to a merchant. Alice establishes her identity with the driver's license bureau, which is analogous to the CA. The bureau issues a driver's license, analogous to the digital certificate.*

The CA is a company or government agency that specializes in issuing digital certificates. The protocol that Alice uses to obtain one and later use it is shown in Figure 8.6. (The driver's license bureau of the physical world could be substituted for the CA without any difference in the protocol.) The three steps are as follows:

1. Alice proves her identity to the CA (for example, the CA might require Alice to appear in person and present her birth certificate, social security card, and driver's license). The CA then

Certificates and a National Identity Card

U.S. citizens have always resisted a national identity card because of privacy concerns. In the physical world, it has been possible to get along without one. However, authentication on the network for general purposes such as electronic commerce requires the equivalent in networked computing—the digital certificate. Are citizens finally ready to accept this?

generates a new unique matched pair of asymmetrical encryption keys on behalf of Alice. The CA gives the secret key to Alice (which she must *keep* secret) along with a digital certificate that contains both her identity and her matching public key.

2. To establish her identity, and also her public key, with Bob (or to anyone else), Alice supplies the digital certificate provided her by the CA.

3. Presuming that Bob trusts the CA, he can use the public key from the certificate not only to authenticate any person claiming to be Alice (by the challenge/response protocol, establishing that the person has Alice's secret key) but also to send confidential messages to Alice.

EXAMPLE: *The most prominent CA is VeriSign, a company that specializes in security and makes money by issuing certificates. Like all CAs, VeriSign maintains strict internal security practices. For example, revealing Alice's secret key to a third party would be an egregious act.*

In security, there is often a catch. What is to prevent someone from making his or her own digital certificate and claiming it comes from the CA? For example, why can't Eve manufacture a digital certificate that attaches her public key to Alice's identity? Alternatively, why can't Eve modify Alice's digital certificate to substitute her public key for Alice's? If Eve can do either, she can fool Bob. These problems can be circumvented by using message integrity.

8.2.3 Message Integrity and Nonrepudiation

Recall that the *integrity* of a message sent by Alice is the assurance that the message was not modified in transit to Bob (analogous to a paper document written in permanent ink), and *nonrepudiation* allows Bob to prove that Alice created the message (analogous to a signed document). Integrity is necessary for nonrepudiation, since without it Alice can always repudiate the message by claiming it was modified after she created it.

Integrity and nonrepudiation can be achieved by using the asymmetric encryption protocol in reverse—encryption with the private key followed by decryption by the public key. Alice can encrypt her message to Bob with her secret key, and Bob can decrypt it using Alice's public key. While this does *not* provide confidentiality, Bob *can* be sure the message was not modified after Alice encrypted it. Why? Anybody can decrypt Alice's message and modify it, but they lack Alice's secret key, which would be needed to re-encrypt it.

This asymmetrical encryption reversal can also prevent Alice from repudiating the message. If Bob keeps a copy of the encrypted message and also a copy of Alice's digital certificate, he can prove in court that encrypting the message required the secret key corresponding to Alice's public key (as verified by the CA)—which could only be done by Alice—and that the message was not later modified by him or anyone else. Alice's only defense would be that her secret key was accidently revealed to someone else before the message was sent, but her case is weak if she did not report this to an authority.

For computational efficiency, a slightly different approach is used to achieve integrity and nonrepudiation, called a *digital signature*. It uses essentially the same idea, so details are omitted.

The security of the digital certificate (see "Digital Certificates and Certificate Authorities" on page 258) also depends on integrity and nonrepudiation. When the CA issues Alice's digital certificate, it attaches its own digital signature. Integrity of the certificate ensures that the certificate was not modified and that it originated with the CA.

ANALOGY: *If the CA is analogous to the driver's license bureau and the CA-issued digital certificate is analogous to a driver's license, its digital signature is analogous to the seal authenticating the license and the plastic encasing of the driver's license that prevents it from being modified.*

The last detail is that Bob requires the CA's public key to check its signature on Alice's digital certificate. This is usually addressed by building it into software.

EXAMPLE: *Web browser software includes the public keys of a set of CAs (and the user can add more). This allows the browser to check the authenticity and integrity of any certificate issued by one of those CAs. The browser can obtain authenticated public keys of Web servers from certificates supplied by those servers and use these public keys to check the integrity of messages, send confidential messages to the server, etc.*

Although all these protocols sound complicated, they are normally embedded in the software applications, so the user need not deal with them directly. The purpose of this discussion has been to understand what security protocols can accomplish and what assumptions are necessary to assume they are trustworthy.

8.2.4 Combining Techniques

Several security capabilities and techniques have been described. In practice, if Alice and Bob interact over the network, they desire a combination of confidentiality, authentication, integrity, and nonrepudiation. The techniques can be combined in different ways to achieve this, as illustrated by the following examples. Assume that Alice and Bob each possess her or his own secret key and a digital certificate issued by a CA.

EXAMPLE: *Suppose Alice wants to communicate a message confidentially to Bob, and Bob insists on authenticating Alice and verifying the message integrity. Alice can encrypt the message twice, first using her secret key (to assure integrity and also authenticate herself to Bob) and then encrypting the result using Bob's public key. She also sends him a copy of her certificate. Bob can obtain Alice's public key from the certificate and her original message by decrypting the message with his own secret key (ensuring confidentiality) followed by Alice's public key (authentication and integrity).*

Suppose Alice wants to send a lot of messages to Bob confidentially but is concerned about the computational cost of asymmetrical encryption. (For example, she might want to actually talk *to Bob confidentially for a while, and her computer or Bob's might not be able to keep up.) What she can do is create a random key for a symmetrical encryption protocol. This key, called a* session key, *will*

be used only once. She can communicate this session key confidentially to Bob using asymmetrical encryption and Bob's public key, which she obtains from his digital certificate. Subsequently, she encrypts the messages using symmetrical encryption with the session key, and Bob and only Bob is able to decrypt it.

This last example illustrates a benefit of a session (see Section 7.2.4 on page 227). It creates a shared context for a sequence of messages, allowing a one-time session key to be communicated to Bob only once and used on many messages.

The possibilities are numerous, but these examples illustrate some important combinations. These techniques are widely used to provide secure communication over the network.

EXAMPLE: *Netscape Communications defined an open de facto standard called Secure Socket Layer (SSL) that adds authentication, confidentiality, and integrity to sessions using the internet's TCP (see Section 7.3.3 on page 235). This protocol is widely used, particularly for secure connections between a Web server (such as retail, brokerage, and banking sites) and the browser. SSL's operation is indicated by a "key icon" in the lower-left corner of a Netscape Navigator window.*

8.2.5 Security Policies

Usually security features are built into software applications and don't disturb the user. However, they can be invasive as, for example, when a user must obtain a digital certificate to participate in a secure application. As a result, strong security techniques are sometimes not used, in spite of the resulting vulnerabilities. To avoid problems that might result, a critical element of security is a set of *security policies* established by an organization as part of a comprehensive plan for achieving a desired trade-off between security and ease of use (see Section 7.1 on page 216). Policies establish, for example, when confidentiality and authentication are required or not required. Policies are enforced primarily by system administrators, who configure hosts and applications in accordance with those policies.

Legal Sanctions

No matter how many security measures are taken, it is possible for an intruder to gain access by surreptitious or fraudulent means. An extreme example would be breaking and entering a physically restricted facility, but there are other ways, such as a confidence game or blackmail. In these cases, the intruder may have broken federal or local laws and can be apprehended and prosecuted.

EXAMPLE: *A particularly notorious individual who gained surreptitious access to many systems is Kevin Mitnick [Haf95], who was convicted of a felony.*

Laws and legal sanctions are the ultimate protector of computer systems.

Access Control

One essential security measure is *access control*, which determines and limits user access to individual applications, to individual hosts, or to entire intranets (see Section 3.3 on page 99). An important security policy concerns access rules based on "need to know" and "need to use" criteria. Access control requires several elements:

- A secure access database describing access authorizations for each user.
- Authentication protocols to authenticate each user requesting authorized access.
- Some way to prevent access to unauthorized users.

This third element is provided by the firewall (see Section 3.3.1 on page 99).

Firewalls

A tool for access control is the *firewall*—equipment imposed on a network link that connects a protected enclave (the intranet) from the global Internet. The firewall is configured to restrict communication passing through, and it offers a focal point for enforcing policies on access to the intranet. Further, a firewall often restricts communication protocols and applications.

ANALOGY: *Countries typically have checkpoints at their border crossings where laws relative to contraband, immigration and emigration, etc., are enforced.*

If hosts were intrinsically completely secure, they could be directly connected to the global Internet, but in practice hosts and their applications have security loopholes. The greatest loophole is the users with legitimate access to the host, who are often not as trained in or conscious of security issues as would be desirable. If an intruder gains access to a legitimate user account, for all practical purposes that host is compromised. Thus, it is better if intruders cannot reach the host at all. Unfortunately, it isn't as simple as a blanket denial of access to all outsiders, because an organization typically wants unrestricted access to some applications, such as

Web servers targeted at the general public. Thus, access control (using firewalls) may be necessary.

ANALOGY: *A country usually restricts its borders but also allows other countries to operate embassies and consulates within its borders. Those other countries may be permitted free passage and unrestricted access to their own embassies.*

An intranet offers no protection against threats *within* the trusted enclave. Most organizations also have internal compartmentalization, restricting access using firewalls *within* an intranet. Firewalls are discussed further in Chapter 11.

8.3 Electronic Payments

Electronic commerce has challenging security requirements because it involves the transfer of money, with the threat of large economic losses (see Section 2.6.3 on page 64). In addition, acceptance by the general public depends on its confidence in the intrinsic security against various threats, such as theft of money and credit card numbers, being stuck with charges made by others, etc. It is wise for a merchant and a customer to authenticate one another—so that the customer is not duped by an unscrupulous merchant, and the merchant is assured of payment—and the merchant obviously also demands nonrepudiation of a customer's order. All these can be achieved using security capabilities already discussed.

A security issue unique to electronic commerce is electronic payments. The ramifications of a security lapse in electronic payments are severe—the possibility of widespread forgery or theft. A customer may be concerned about privacy and thus desire confidentiality in his or her purchases. This requires, among other things, that purchase information not be made known to anybody but the merchant and that the customer's financial information remain with the financial institution. These issues place challenging requirements on an electronic payment system.

In the physical world, payments are made by cash, credit cards, debit cards, and personal checks. Credit and debit cards can be

used on-line, presuming appropriate authentication and confidentiality. Electronic funds transfer from one bank account to another can also be used, which works like a check. There are, however, reasons to consider innovative alternatives:

- It may be possible to improve user convenience.
- Added security measures can reduce the vulnerability of all parties to fraud.
- A credit card payment, having a substantial transaction cost, is not economic for small purchases. Particularly in the sale of on-line information, there is a desire for small payments, (like fractions of a cent, often called *micropayments*). Thus, payments with small transaction costs are desirable.
- Many consumers are concerned about privacy and don't relish any ability of merchants or financial institutions to track these purchases.

There are two basic approaches to payments, whether electronic or not:

- *Account authorizations* linked to purchases (such as credit and debit cards), in which payment is transferred from financial institution to merchant, drawing upon the customer's account (a debit) or as a loan to the customer (a credit).
- *Tokens of value* carried by the consumer, such as cash, stamps, and increasingly in the future, something called a *prepaid smartcard* or *digital cash*. These tokens of value have intrinsic monetary value (either directly, like a coin, or indirectly backed by a financial institution or the government, like a bill) and can be exchanged for goods and services. They can be stored in a card (similar to a credit card) or on the disk drive of a desktop computer. Payment is direct from consumer to merchant by transferring the appropriate token(s) without an intermediary.

EXAMPLE: *A postage stamp is a token of value that can be exchanged for a single specialized postal service. It is affixed to the item being mailed.*

8.3.1 On-Line Credit Card Systems

Credit cards, today common for payments by telephone or over the network, work well for relatively large payments. They also have problems:

- Consumers are concerned about the theft of credit card numbers. This theft can occur by eavesdropping (which is countered by confidentiality, such as the Secure Socket Layer used by many electronic commerce Web sites) or by merchant employee theft.
- Consumers are concerned about privacy. Each merchant can track the purchases of a given consumer (because the same credit card number is used each time), and the financial institution can track all purchases made on a given credit card.

These concerns are addressed by an electronic credit/debit payment system called SET.

Secure Electronic Transaction (SET)

SET is a standard for on-line credit/debit card transactions established by Visa and Mastercard. In a conventional credit card transaction, the consumer deals with the merchant, and the merchant deals with the financial institution. The consumer provides credit card information to the merchant, who both authorizes the charge before fulfilling the consumer's order and submits it to the institution for payment. A key shortcoming addressed by SET is the availability of the credit card information to the merchant, which enables the tracking of purchases and increases the possibility of theft of credit card numbers.

SET logically partitions the transaction into the order and the payment authorization. The *order* for goods and services—and fulfillment of that order—is conducted between the consumer and the merchant. The *authorization for payment* is conducted between the consumer and the financial institution, which credits that payment to the merchant. This partitioning ensures that the merchant has no visibility of credit card or financial information, and the financial institution has no visibility of the order. The protocol has all the previously discussed security features, including authentication of all parties and confidentiality.

SET includes two more subtle features. First, the authorization actually passes through the merchant, so that the consumer has the appearance of dealing only with the merchant. This authorization is encrypted, and the merchant does not have the secret key necessary to view it. Of course, the merchant does obtain confirmation from the financial institution that it will receive payment before order fulfillment. Second, the order and authorization are linked so that the customer cannot later repudiate the purchase, for example, by claiming that the payment was for other goods never delivered. This uses a variation on the digital signature, called a *dual signature*, which allows both merchant and financial institution to verify (and prove) the linking.

Digital Cash

Another on-line payment mechanism is *digital cash*, which is a token of value in networked computing—analogous to bills and coins in the physical world—that can be directly exchanged for goods or services. There are many nontechnical issues raised by any new token of value (see the sidebar "Questions about Digital Cash"). There are also interesting technical issues.

Like everything in networked computing, a token of value in a digital cash system must be represented by data; that is, a string of bits. Presumably, this data includes the monetary value, as well as possibly other things. This raises some troubling questions.

First, how can somebody accepting digital cash be convinced of its monetary value? The answer is it must have an issuer (such as a central or financial institution) that stands behind it. The issuer must not be able to repudiate it, and the merchant must verify its integrity. A consumer will obtain digital cash by taking an account withdrawal in digital cash. (Financial institutions love this idea, because to them it is an interest-free loan!) The digital cash carries a digital signature of the issuer, assuring its integrity (it cannot be modified by anybody else without detection) and also precluding the issuer from repudiating its value.

Since digital cash is data, unlike coins and bills it is easy to replicate. What, then, is to prevent the consumer from spending it more than once?

Questions about Digital Cash

New tokens of value, such as digital cash, raise a number of questions.

One obvious one is who will back the value. Is it backed by the financial institution, as in a demand deposit withdrawal, or is it backed by the Federal Reserve Bank, as in paper money, or is it backed by somebody else? What happens if a token is lost or stolen? Can the value be restored, or is it irrevocably lost? Can the thief spend it, and if so can the thief be traced? Another issue is whether digital cash is subject to regulation and if so by whom and for what societal purpose. For example, value tokens add to a nation's money supply, the size of which is tracked and controlled by a central bank to preserve monetary value. What is the impact of the global reach of the Internet and electronic commerce? Who pays for the underlying infrastructure and operation of a digital cash system, and how do they recover those costs? Is it the government recovering costs through taxation (or selling the cash they manufacture), or financial institutions through fees? Does digital cash replicate the anonymity of coins and bills, such that purchases cannot be traced?

EXAMPLE: *Unless countermeasures are taken in the digital cash protocol, it would be easy to withdraw $1 from the bank, make 999,999 copies of it, and then make a $1,000,000 purchase. Naturally, the bank must ensure this cannot happen.*

This problem is circumvented by including a unique *identifier* in each token. The identifier is a big number, just like the serial number on a physical bill. A replica can be detected by comparing identifiers. The question is, where and when is the identifier checked? The issuer can detect multiple spending by refusing to deposit a token, when a token with the same identifier was previously deposited. To enable this double-spending detection, a token can be spent only once, and then it must be returned to its issuer for credit to a bank account (or a new issuance of digital cash). Digital cash is thus different from physical cash in that a merchant accepting payment by digital cash cannot spend that token again—it must be deposited in the issuing institution.

Another issue raised by the unique token identifier is the possibility of tracing of financial transactions, with privacy implications for the consumer. This issue is addressed by *anonymous* digital cash (see the sidebar "Privacy and Anonymous Digital Cash"). It preserves the spending anonymity of physical cash; namely, when a merchant receives physical cash for payment or a financial institution receives physical cash for deposit, there is no record of who spent the cash or what they bought.

If an institution can match identifiers of tokens it issues against the identifiers of tokens it accepts for credit, it has a complete trace of how the cash is spent, in light of the single-spending attribute. Fortunately, while the identifier must be unique to prevent multiple spending, it need not be known by the institution when the token is *issued*. The key idea behind anonymous digital cash is to hide the identifier from the financial institution at the time of issuance. While this requires enormous ingenuity, it has been shown to be possible.

Privacy and Anonymous Digital Cash

There is a tension between the desire of consumers for privacy in their purchases on the one hand, and a societal interest in preventing tax avoidance and money laundering on the other. The latter encourages audit trails for monetary transactions. Already, with the level of credit and debit card purchases, a great deal of information is becoming available about the spending patterns and lifestyles of many consumers. If in an electronic commerce system all purchases are logged and traced, the private lives of individuals would be an open book to financial institutions. For example, if every purchase of gasoline, payment of toll on a highway, and minor purchases of snacks and drinks were logged, it would be possible to effectively track the location of individuals almost as effectively as by surveillance.

David Chaum is the most eloquent and persistent advocate of privacy in electronic transactions and has developed protocols for anonymous digital cash. To quote him [Cha91]:

> We are fast approaching a moment of crucial and perhaps irreversible decision, not

(continued)

8.4 Open Issues

Trustworthiness is arguably one of the least mature areas of technology and one that raises contentious policy issues.

merely between two kinds of technological systems, but between two kinds of society. Current developments in applying technology are rendering hollow both the remaining safeguards on privacy and the right to access and correct personal data. If these developments continue, their enormous surveillance potential will leave individuals' lives vulnerable to an unprecedented concentration of scrutiny and authority.

8.4.1 How Do We Deal with Increasing Vulnerability?

Networked computing integrated into vital societal functions carries vulnerabilities. Even technological security, such as that discussed here, cannot by itself be completely effective. Physical security, employee integrity, operational vigilance, and many intangibles are involved. Further, all possible threats cannot be anticipated, so the update and addition of security measures throughout the life cycle of an application is inevitable.

Most unsettling is the observation that network breakdowns or break-ins can be massive, with much wider impact than physical breakdowns or break-ins. Further, the magnitude of the risks is difficult to quantify, and as a result the insurance industry has not been active in offering means for spreading these risks as it has in other human activities.

How will these problems be addressed? One clear need is additional research, seeking more effective security systems and greater robustness and reliability. Another need is for a much more vigorous response on the part of government and law enforcement, especially in developing more effective means for deterrence, detection of illegal or harmful activities, and investigative techniques.

8.4.2 National Security and Law Enforcement Needs

Government policy and laws applying to cryptographic technology must make some difficult trade-offs, and this is an area of strong disagreement between governments and industry [CST96a].

On the one hand, strong encryption technology—the basis of all the security tools discussed—is extremely important to networked applications such as electronic commerce. A credible threat to any country is economic espionage, which can be countered by maintaining confidentiality in commercial network traffic. The government has a law enforcement obligation to counter domestic and

foreign threats to commerce and industry, and thus should encourage encryption technology. On the other hand, the government relies upon electronic eavesdropping to address legitimate national security threats and court-ordered wiretaps in criminal law enforcement and prosecution. These require a government to decrypt intercepted messages, which motivates it to keep "strong" encryption away from criminals, terrorist organizations, and foreign governments. (Here, "strength" is generally related to the number of bits in the encryption keys.)

One strategy employed by the U.S. government has been to legalize any and all encryption within the country but to disallow the export of strong encryption technology. Industry complains that this not only makes U.S. software less competitive globally but also stimulates viable overseas competitors. A U.S. government response has been to propose "key escrow" schemes that allow a back door for "authorized" access to encrypted information by forcing keys to be deposited with an escrow organization under legal obligation to disclose them in national security and criminal cases. Industry has found these proposals unsatisfactory, arguing in part that individuals and businesses should be afforded the strongest protections available in the technology. Also, these proposals are difficult to implement globally, across multiple jurisdictions. These policy issues are far from settled and will doubtless be debated for some time.

8.4.3 Individual Privacy

The degree to which individual privacy will be maintained in electronic commerce remains an open issue. Commercial enterprises have incentives to track the purchasing habits of their customers in a legitimate attempt to improve their business strategies, but consumers worry that this information can be misused. Similarly, an increasing number of information access applications on the Web are supported by advertising, and the more information available about users, the higher the advertising rates. This tension between the interests of sellers and information providers on the one hand and consumers and users on the other has an uncertain outcome.

EXAMPLE: *The Open Profiling Standard and TRUSTe are industry initiatives to make visible to the consumer where the self-provided information is used and to lend some control over personal information. An objective is to reassure consumers, so that privacy concerns don't become a major impediment to consumer acceptance.*

8.4.4 Theft and Piracy of Software and Information

A concern on the part of sellers of information and software is the ease of replication and unauthorized dissemination over the network. Security measures such as encryption provide sellers some tools to enforce license provisions. However, there are limitations, since information must be decrypted in order to be read or viewed, opening opportunities for copying if not outright piracy. Like other security measures, they are also inevitably invasive to the consumer. Some strategic issues for sellers are discussed in [Sha98].

Further Reading

An excellent overview of the various compromises to trustworthiness is [CST98], and the implications for essential societal infrastructure are summarized in [OST97]. [Ber97] gives a good practical picture of computer system reliability. [Sch96] is an excellent compendium of security protocols. Cryptography policy is considered in great detail in [CST96a].