Computer security - a layperson's guide, from the bottom up

Karen Sparck Jones
Computer Laboratory, University of Cambridge
William Gates Building, JJ Thomson Avenue, Cambridge CB3 0FD, UK
sparckjones@cl.cam.ac.uk, http://www.cl.cam.ac.uk/~ksj

June 2002, Version 1.2; Computer Laboratory Technical Report 550

Computer security is about several things. Thus security is
(1) stopping outsiders destroying systems by virus or denial of service attacks;
(2) stopping invaders perverting systems by hacking into controls or data;
(3) stopping insiders using systems in ways their designers did not intend.
This note is primarily about (3). It is not about (1), but in large and amorphous organisations (like governments) doing (3) properly means attending to (2) as well. Security in sense (3) depends on policy. The purpose of this note is to show that effective security policy requires attention to computational realities, i.e. to what sorts of things can be done by technical means and, more importantly, to what cannot. Computer security depends, critically, on people, and the principles involved are independent of technological detail.

Introduction

A great deal of information about people is held, manipulated and communicated in computer-based systems. Information privacy, confidentiality and integrity matter to people. *Computer security* is the means for supporting proper information states and operations and for guarding against improper ones. Computer security is thus a very broad notion, with pervasive application to computer systems and correspondingly large implications for security engineering. Maintaining privacy and confidentiality for information about people, or particular to people, is one element of computer security. However it is essential not to focus on these alone, since the way they are implemented follows from the way security in general is implemented. Both formally and technically the mechanisms for protecting information that individuals may regard as their private information may be, and usually are, the same as the mechanisms for protecting the information intended only for some particular process within a larger system. For example, the technical means for preventing improper access to someone's personal medical data within a system dealing with different aspects of hospital administration may be exactly the same as those used to stop one part of a system running a chemical plant from picking up numerical data intended for another. Thus as this comparison suggests, talking about computer security implies *both* that personally important notions like privacy and confidentiality have to be independently defined, *and* that the way these notions are to be interpreted in the context of computer capabilities and constraints has to be addressed.

There are many common beliefs about computer security that are wrong, for instance:

1. some particular technological gadget - SSL, or PKI, or whatever, is sufficient to ensure security that meets some generic application needs, e.g. SSL will fix up e-commerce;
2. even if current gadgets aren't good enough, further work on encryption is all that's needed because we'll get perfect encryption before long and then we'll be able to wrap all our communications and data up in it and there won't be a security problem;

3. software comes in whole packages with built-in security, to check user access from the outside, or to stop users when inside from going into particular places they oughtn't to be, so as long as you buy your software from a reputable producer everything will be OK;

4. hardware comes in boxes which can have heavy-duty casing round them or tamper-proof seals to stop people getting into them and e.g. interfering with the communication links;

and so on.

These are all misconceptions grounded in the belief that security is only about computing systems in themselves, and nothing to do with the humans who design, build, use, and manage them, which is in fact where all the real problems are. Security is primarily a human problem, to do with the contexts in which systems are developed and used, and only secondarily a purely technical problem that has to be, and can safely be, left to the professionals.

Computer security has many deeply technical aspects ranging from the way chips are made to the use of number theory in cryptography. However the essentials are perfectly ordinary and familiar notions. These are therefore deliberately laid out without any technical apparatus, in Part 1 of this note. The situations used as examples in Part 1 may seem trivial and hence unproblematic. But analysis shows that such simple cases can be much more complicated, from a security point of view, than they appear. Security is a complex matter, even without computation. The examples in Part 1 are intended both to illustrate the key points that apply to security, computational as much as non-computational, and to provide the background for the specifically computational manifestations of security treated in Part 2. Computational security can be quite powerful, but can also be very complicated, with ill as well as good effects. Part 2 provides some (though still modest) technical detail, to show what security looks like in computational systems, and to provide an explanatory context for frequently-encountered security jargon, for instance 'public key encryption'. Tools like public key encryption are valuable means of maintaining computer security. However they are not panaceas, and security in computational situations requires just as careful and detailed analysis as in non-computational ones.

Note: technical terms are not universally agreed in the trade; this note seeks to follow reasonably established practice.

Part 1 : Homely examples

I start with homely examples: but they illustrate all the essentials for computer security. I will therefore present these examples in some detail, to provide appropriate hooks for attaching specifically computational points later.

Access to a store

We have a store, with a lock, containing some goodies. We have an opener for the store, operated by an agent.

```
EXAMPLE SETUP 1 :

    Store is a wooden chest with an iron Lock
    Goodies is a pile of ducats
    Opener is a big iron device
    Agent is a human
```

The assumptions are that the opener fits the lock, and that the opener when turned in an obvious way opens the store.

As a mechanism for *protecting* the goodies this setup has *weaknesses*, including

1. there is no guarantee that the agent is authorised
2. there is no guarantee that the opener is unique
3. there is no guarantee that nothing other than the opener opens the lock
4. there is no guarantee that the store is robust.

Thus for instance anyone who can get at the opener can open the store, or someone with a copy of the opener can, or possibly with a skeleton opener; as described, the store might not be strong enough or heavy enough to resist being broken open or removed for more serious and leisurely attack.

Also, considering the situation from the point of view of *access* for the agent rather than protection for the goodies,

5. there is no guarantee that the lock will actually work
6. there is no means of recovering from loss of the opener
7. there is no certainty that the store is what it seems
8. there is no provision for knowing where the store is if it is moved.

Thus the lock might get rusted up or warped, the opener could be mislaid, the store might be replaced by a copy not identified as such for some time, the store might be legitimately moved but without record.

These are only some of the generic weaknesses in Setup 1, and some possible illustrations of the specific forms they may take; but they are enough for now.

Since it is possible that existence of the goodies may be known or suspected, some of the weaknesses leave the goodies exposed to *threats*, e.g.

1. the opener could be stolen, or copied
2. the store could be stolen.

From the agent's point of view, someone else using or substituting for his opener could be viewed as impersonation and regarded as a threat.

Other weaknesses e.g. natural failure of the lock, are not threats. But to ensure protection for the goodies and access for the agent, it is proper to do a *risk analysis* to find out how likely they are to occur and a *loss analysis* to work out what the consequences of such a failure are.

For example, the risk of lock failure may be low and the cost of fixing it to ensure access to the goodies calculated as low (and the risk of not being able to fix it be judged very low). However while the risk of opener duplication might also be judged low, the consequent risk of loss of the goodies could be high and cost of loss of the goodies also be high.

The response to the analysis of the situation is some security *procedure*, for example to keep the opener carefully locked up in some place known only to the agent, who carries this subsidiary opener on their person, to keep the lock oiled, etc. Note, however, that having created a security procedure in response to the initial situation, it is essential to do a further security analysis, or *audit*, for the revised situation, e.g. what happens if the subsidiary opener is lost or stolen?

This example, even though given with some elaboration, still omits much important detail e.g. how many other people than the agent know that the store and opener exist and where they are; just how robust the store is, e.g. is it very heavy indeed, bolted to the floor, etc; how large and irreplaceable the goodies are. One of the important points is whether the agent who is required to actually open the store is also the person we can call the beneficial owner of the goodies, or is some intermediary.

Thus any particular version of Setup 1 would need much more exhaustive and careful *description* than that sketched, and thus a much fuller analysis of its weaknesses and threats, and risks and costs. For example, suppose the goodies are very valuable or very bulky, and are thus divided into subsets in different places: this obviously implies a much more elaborate *scenario* for the entire situation. For instance, should there be different openers for each of the now separate stores or just the same one for all?

It is evident that whatever security procedure is adopted will have to make a safety/convenience tradeoff. Thus e.g. having several stores in separate locations with different openers also kept in separate places reduces the risk of losing all the goodies, but increases the management hassle of keeping control over all the openers and ensuring all the different locks are oiled.

It is particularly important to consider the implications when the agent is not the same person as the beneficiary, e.g. the beneficiary may be unable through illness to go and get the goodies and have to employ someone else as their agent. The beneficiary may think they have sufficient reason to *trust* their agent unreservedly and simply hand the opener to them and tell them to use it on the store. Or they may decide to have a more explicit delegation or handover subprocedure, i.e. a security *protocol*, e.g.

1. make the agent sign the borrowing book for the opener
2. hand the opener to the agent
3. tell the agent where to go for the store, what goodies to take, etc
4. receive the goodies and the opener from the agent
5. get confirmation from the agent that the store is in place and is locked
6. complete the signoff for the agent in the borrowing book.

Clearly, there has to be some *trust* in the agent even in this less informal case. Part of the security analysis for the particular version of Setup 1 in question is determining whether the safety/convenience relationship is reasonable.

More importantly, the Setup 1 example emphasises two fundamental points about security:

FP1: *There are always humans somewhere in a security situation, or there wouldn't be any point in having the security.*

FP2: *There have to be compromises between protection and access, since complete safety precludes access and universal access nullifies safety.*

FP1 applies even where the beneficiary and the agent are identical, since the beneficiary's interests over time are not necessarily well served by his own agency (e.g. he forgets where he put the opener). However since most situations involve more than one person, FP1 clearly has vital security implications. The need for compromise in FP2 is manifest, e.g. it is not obvious what locking the goodies up in the store and throwing the opener away achieves for the beneficiary.

Access to a person

Setup 1 had a passive physical object, the pile of goodies, as one side of the relationship. Now consider the case where access is to a human. We start by considering only a scenario for making a connection with the human.

Thus we imagine we have a room with a telephone, T1, in it on the human's desk and an external door with another telephone, T2, by it. The simple model is the entryphone one.

```
EXAMPLE SETUP 2 :

    Room is an office with a telephone T1, an entryphone inside-end
    Human is a person in the room
    Door is on the street with telephone T2, an entryphone outside-end, by it
    Agent is a human at the door
    'Enter' is an action that opens the door
```

This Setup is clearly analogous to Setup 1: the room is equivalent to the store and telephone T1 to the store lock, with the human in the room equivalent to the goodies in the store. The telephone T2 is equivalent to the opener, and we have an access-wanting agent as before. However it is necessarily a bit more complex because there has at least to be some means for the human in the room to react to a ring on telephone T1 and open, or allow the agent to open, the door. This is the 'Enter' action, realisable in various ways e.g. the human descends and opens the door, he presses a button in his office to release the catch and the door swings wide, the agent is sent the instruction to open the now released door.

But now if we consider Setup 2 as described from the point of view of protection for the human - i.e. as ensuring that only proper people should reach human's room, - it has clear weaknesses. These include

1. anyone can use telephone T2
2. there is no guarantee that the person who used T2 is the same as the person entering.

Thus even a rather simple security analysis for the human suggests that more procedure is required, and indeed one can imagine that the agent might also be interested in establishing that it is really human that they are going to see. One possible way of dealing with human's concern is to require that agent additionally submits a numerical code or password when using their telephone T2. However, this obviously also requires that human be satisfied that the password is only given to persons that human is willing to see, and this in turn involves a further *authentication* subprocedure.

An alternative approach, which might also go some way to meeting agent's concern that they will really be seeing human, is to allow for some interaction between human and agent designed for mutual authentication. The minimal form this could take would be to have passwords offered by both parties. However this makes for a more complex authentication requirement, and it might appear that a more natural interactive dialogue might be more satisfactory, e.g. both agent and human provide substantive information about themselves, relying on background world knowledge for plausibility assessment.

However, there are still weaknesses in such a scenario. Thus, for human, these include:

1. there is the possibility that agent is an impersonator
2. agent is not an impersonator but is acting under duress by an unknown,

while from agent's point of view

3. there is no guarantee that human is not similarly suspect.

Note also that the situation as described, when considered in detail, has other weaknesses, e.g.

4. there is no provision for ensuring that the door is closed when agent is let in (or out).

Thus as before, it is necessary to complete a risk and loss analysis, e.g. of the agent being an impersonator and what the human has to lose if so (e.g. if the human constitutes goodies, and pseudo-agent kidnaps human, what will this cost real agent)?

Richer scenarios

Both Setup 1 and Setup 2 were necessarily anchored in physical location, Setup 1 by using a chest, Setup 2 by involving a door and office. We can clearly envisage the case where the goodies in Setup 1 are not physical but are abstract, e.g. some data, facts or information, and where it is not necessary for the agent to know anything about where the goodies are held. However this emphasises the problem of whether the store is genuine, and of also whether the goodies are what they seem. We already allowed under Setup 1 for the possibility that the store was a fake, and we also *assumed* that the agent would recognise ducats, that they would be genuine ducats, and also that the agent (or beneficiary) would have some independent knowledge of how many ducats there should be. However once seeing with one's own eyes, albeit perhaps unreliably, is removed, establishing that the store and the goodies are what they ought to be is a more challenging issue.

Thus suppose we have

```
EXAMPLE SETUP 3 :

   Store is a library service with a dialup as Lock
   Goodies is a financial news summary
   Opener is a dialup number
   Agent is a human.
```

In this situation, unlike Setup 1, protection for the goodies is not immediately at issue because the goodies do not belong to the agent (or the beneficiary). As described, there is no need to establish that the agent is authorised, and so forth. However from the agent's point of view the weaknesses include

1. there is no check on whether the dialup number actually goes to the store
2. there is no guarantee that the goodies are genuine

while as before, if the beneficiary is separate from the agent, the beneficiary cannot know whether the agent actually contacted the store or just made it all up.

Clearly, if we pursue the idea that some other human than the agent/beneficiary has an interest in the goodies' status and integrity, we get a richer and more symmetrical situation. For example if we now have a human as the person responsible for the store and goodies, they have security concerns e.g. that the goodies are not going to be stolen by the agent and used to run a rival service. This might imply some requirement that the agent is a legitimate user of the store and has also, perhaps, has undertaken not to misuse the store and its goodies. Any scenario along these lines would require more detailed description and analysis: thus for instance there has to be not merely a means of authenticating the agent as who they claim to be, but of capturing their undertaking on behaviour (since the goodies cannot be protected *in themselves* once they been given to the agent as information).

Similarly, if we consider a version of the earlier Setup 2 where there is no physical door and no need for the human and agent to meet and no presumption that they already know one another, and where we do not limit ourselves simply to the agent's access to the human, but allow the human to give something to the agent, we have a situation much like the one just considered. Thus - and simplifying by referring just to (human) agents A1, A2, ... - we now have

```
EXAMPLE SETUP 4 :

   Store is a reference service with a telephone T1
   Information is (valuable) financial data
   Agent A1 is a human responsible for the store, and the information,
    with use of T1
   Opener is a phone number for the store
   Agent A2 is a human user of of the store
   Telephone T2 is available to A2.
```

Then, as indicated, we have a number of weaknesses including

1. protecting the information against intrusion
2. insuring the store against being unavailable
3. guaranteeing that agent A2 is legitimate
4. certifying that the store and the information are genuine (to A2)

that a full security analysis would have to address. Further, if we imagine that the information is supplied not only by agent A1, but also by agent A2, say via recording to the store, we then have a further weakness, namely

5. protecting the information in the store against damage,

interpreting damage to cover a range of possibilities, e.g. overwriting original recordings, supplying conflicting data likely to confuse other users etc, which could occur either without evil intent or through malice. Further, for example, depending on the detailed character and value of the information, there is a weakness in

6. guarding the information against ripoff by agent A2.

Secrets and encryption

In Setup 1 we allowed for the possibility that other people might know about the store chest or its goodies contents, as a threat one might wish to avoid. In a general way, there are *secrets* to be kept. If agent and beneficiary are the same, and agent keeps their counsel, the risk of someone else getting at the goodies is fairly small. However if, as was tacitly assumed, the agent is not in the same place as the store and so has to go and fetch the goodies and bring them back home, it is clearly desirable not to advertise the existence of the goodies on the journey. One can imagine the agent concealing the ducats in a modest shopping bag. Whether or not one should conceal the existence of the chest, on the other hand, is not obvious: there is no general answer to what should be a secret.

We can similarly see that where the goodies are not concrete ducats but abstract financial information but this is valuable in its own way, as in Setup 4, it may be appropriate to *encrypt* it en route to its user agent A2 to ensure that it does not also reach other eyes en route. Indeed it may be sensible to keep it in the store in encrypted form as well to guard against some unauthorised person breaking in and reading it there.

Further, with Setup 2, we allowed for the need to authenticate the agent seeking entry to the human in the office, and imagined the agent using a password for this purpose. To avoid others overhearing this when the agent uses telephone T2 and then using the password to gain improper access to the human, the password might be disguised (encrypted) in some way, e.g. as a form of greeting.

Clearly, similar possibilities for having encrypted passwords could apply to Setup 4.

However using encryption adds complexity to the scenario, and thus in turn has to be subjected to security analysis. For example, how easy and likely is it that unauthorised agents will be able to crack it, and how costly will the consequences be, e.g. if a small number of unauthorised people gain financial information?

Privacy and confidentiality

The setups presented so far illustrate security protection for information for a variety of good reasons, but without any particular reference to *privacy* or *confidentiality*, where in the narrowest case the former refers to my keeping information about myself secure and the latter to my keeping information about others secure, but where more generally achieving privacy e.g. for *my* medical data imposes a duty of confidentiality on *others* dealing with them. The only assumption in the example setups has been that protection is required because the goodies involved are of value to someone and should thus not be open to corruption or misappropriation. The corresponding security procedures have thus also involved secrets, but in a quite neutral sense not involving personal privacy or confidentiality.

However it is easy to see that situations where information that would be viewed by those involved as private or confidential, for example, a draft job application or referees' comments on job applications, could figure in setups like those examined. For instance, in the job case we could have a filing cabinet with the comments to which we need to restrict access to certain agents and also use to certain modes for the different agents e.g. to deposit and read, or to read or, at a different time, to remove for shredding. All of this presupposes means of validating these agents and controlling their actions, as well as excluding other agents, as well as ensuring the file comments are not accidentally as well as deliberately altered.

Policies and practices

Doing a proper setup, i.e. situation, description and a security analysis for this, implies work at 2 levels, those of security *policy* and security *implementation.*

Thus, for example, it is a matter of policy that the humans involved are treated only as individuals, or are treated as having *roles*, independent of who they are as individuals. Even in a very simple case this is helpful because it makes it possible to separate the requirements for role holders from the requirements of individuals as candidates for roles. For instance for Setup 2 we may require any human to be a member of some organisation but any agent only to be an adult, and then keep one list of people with their various properties rather than two lists for humans and agents respectively. This last is a matter of implementation for the security policy, since there have to be safeguards about who is on the list and checks that they have the required property.

Having a policy about roles makes it easier to maintain a useful distinction for security purposes between agents that are *authorised* for some function and determining whether an agent, or by extension an action by an agent, for example requesting information, is authentic, i.e. is what it purports to be.

However, all aspects of a setup are within the scope of a security policy, and equally will normally be open to different implementations with different risks. Thus if it is policy that valuable goodies in the form of information shall be protected against accidental or malicious damage, i.e. shall have their *integrity* assured, and that this will promoted by duplication so if one is damaged the other can make it good, what risks (more points to guard) and costs (of copying) follow?

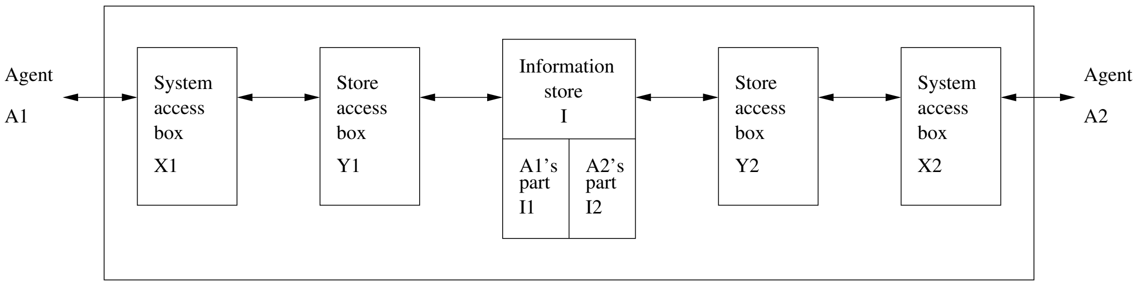
General model

We can now, as the base for examining *computer* security, consider access to, and transmission of, information in terms of the generic functional modules involved, and the steps in the whole communication process. First, computer security has to be considered in its own right. It should not be considered just as a byproduct of the fact that information handling and agent communication are automated for good, independent reasons. Second, computer security has to be examined for its own sake because of the intrinsic properties that computational systems have. In (modern) computational systems security can be violated in an instant and with extremely rapid and pervasive effects, so system design implications for security have to be very carefully examined. At the same time, computers offer particular means of realising security functions in ways that are not merely simple analogues of non-computational ones: computer power means that security functions can be carried out in ways that are not feasible otherwise, e.g. through complex encryption algorithms, which imply distinctive properties in computer security regimes. These are advantages that can make computational systems more secure, though this is not guaranteed.

Computer security can make many individual entities and processes within an entire setup relatively safe. However because setups involve people, i.e. computer systems are not autonomous, security cannot be guaranteed. Further, while even apparently simple setups are really quite complex from a security point of view, many setups are intrinsically complicated since they involve many entities and processes, so the danger of security failures may be quite high.

The setup examples introduced earlier did not involve any computers. It is possible to have all kinds of mixes of computational and non-computational elements in a setup. For convenience we suppose that the goodies we are interested in are always information that is computationally held and transmitted, and that the systems we are dealing with are automated, i.e. computer-based, ones.

We can envisage the essential setup and system model, simplistically, as follows:



This structure covers a wide range of detailed situations including ones where the information store functions as a primary data repository (analogous to a library) and ones where the information store contains no more than the secondary routing information needed to connect agents (analogous to a phone directory); situations where one agent is concerned only to interact with the information store and not directly with another agent; situations where the communication of information is essentially unidirectional, etc.

Of course the model also generalises to multiple agents, multiple stores, multiple boxes, and hence multiple channels.

However for this note, the simple model is sufficient to demonstrate the many points where security enters. Thus X1 and X2 have to have grounds for accepting or rejecting interaction with A1 and A2, and also interaction with Y1 and Y2. The analogous situation applies to Y1 and Y2 in relation to the modules on their two sides. Finally I has to have grounds for interacting with Y1 and Y2, and perhaps also grounds for allowing internal interaction between I1 and I2. These grounds may refer both to which module is engaged in the interaction and also to any indirect third parties, and may refer both to what the interaction is about, i.e. *message content*, or who is seeking it, i.e. *message producer*. As this implies, the various boxes apart from the information store may have their own data stores to enable communication control. Thus there are potentially many information points that have to be kept secure. Equally, the communication channels between boxes have to be kept secure.

All of this may seem absurdly elaborate. A simple example illustrates the need to recognise how real such complexity is and thus how pervasive the need for security is.

```
EXAMPLE : MEDICAL DATA BASE

 A1 is a ward sister, A2 is a medical researcher.
 A1 and A2 are both users of patient record data, at I.
 A1 in addition has ward log data in I1, while A2 has epidemiologial data in I2.
 For A1 to access I, X1 has to accept her logon as a system user
  (this could legimately be for other purposes than to reach I).
 Y1 has to accept X1's contact, and A1's access to I.
 Similar considerations apply to A2 via X2.
 I1 has to accept A1 for reading and writing data.
 I2 can accept A2 for reading and writing data, but exclude A1.
 I1 can also accept A2 for reading data only.
```

All of this depends on a secure creation, entry and transmission of, A1 and A2's individual authentication i.e. passwords, on secure authorisation of A1 and A2's status with respect to I, I1 and I2, also on secure authentication and authorisation as participating automated agents between X1/Y1 and X2/Y2, also on security (and integrity) of the authentication and authorisation data, though these processes and data serve only to support the mechanics of A1 and A2's travels through the system. In addition, for the system to support its primary functional task as a medical information system it is necessary to support the security (and integrity) of information in I, I1 and I2, and in its transmission to and from A1 and A2 in reading and writing respectively.

Moreover, as even this rudimentary sketch suggests, security has to be regarded as a *systemic* matter applying to an entire organisation, which depends critically on the extent to which the security policy is made to stick throughout the organisation, not just in the strictly computational parts. For example, people can be forced to adopt new passwords by machines blocking access every month, but this in itself is not enough to stop people choosing passwords that are relatively easy to guess. At the same time, as this illustration suggests, while new technology, especially computing, appears to offer more security power, the fundamental principles of security are independent of technology, and it is easy with technology to create complexities that harbour damaging design failures or implementation bugs.

Part 2 : Some technical amplification

It is evident that implementing computer security has many technical ramifications as well as human management requirements. In Ross Anderson's comprehensive *Security Engineering*, the central technical areas include authentication and passwords, encryption, system access control in general, security management in distributed systems and networks, and in systems with multiple different security regimes, with file and database protection logistics as essential to all of these. As already mentioned, even these interact with human factors, e.g. people's inability to remember long passwords or PINs, and the human role in security policy formulation and in system design, management and evaluation is obvious.

The general presumption in Anderson's book is not just that security can be compromised by acts of God, or the carelessness of man, or by plain old Murphy, but that security systems have to be based on the principle that they will be attacked, not only by criminals out for gain, or by vandals, but by ordinary people who put their convenience first. Security imposes costs on individuals that they would prefer not to bear, and which they regard as unnecessary by some plausible piece of self-justification ('I know Bill would let me look at his files if I asked him, so I'll not bother him but just get in there').

In this part I will consider four technical areas - authentication, encryption, system structure, and database administration, in some, but still minimal, technical detail. My aims are to provide a context for the security jargon that is so frequently encountered e.g. PGP, Kerberos protocol, RSA algorithm, Bell-LaPadula model, and CORBA, and to emphasise that all the technology has its limitations, so it is always worth asking to be shown precisely where the Emperor's new clothes are. Anderson's book provides splendid, detailed further amplification with many real-life examples.

Authentication

Authentication minimally determines that an agent (human or system) is who they say they are, but frequently also checks for legitimate time and place. Authentication is therefore based on a protocol.

I shall start with what by the standards of the trade is a very simple protocol. It is more complicated than might be expected, so the security motivation for the various features calls for comment.

Protocol 1

> A simple protocol for human user access to a system facility, file ... would be that the user submits their identifier along with their password at a system entry point, which may be physically separated from the main system and not just a logical front end, combines the password with a nonsense string, or 'nonce', encrypts this combination using an appropriate *key*, and passes the resulting combination to the facility controller. The controller checks that the identifier is a known one, decrypts the password-nonce combination using the key, checks that password is recorded as associated with the identifier and that the nonce has the required property (see below), and lets the user into the facility by sending a suitable enter signal.

The reason for having a separate identifier and password, and not treating the password as identifier, is that for many system management purposes it is desirable to recognise that there is a single entity - the authorised user - who may have many passwords, for different things or times, and further to allow the identifier to be public. The reason for encryption is that if the entry point is separate from the facility control (as with an ATM and a bank) it is important that an eavesdropper cannot listen in and pick up a password. (The form of encryption is a separate matter: we can pretend here it is something with as simple a key as a letter substitution table

- though for technical reasons this would not work in this case, - which might be common to all users or individualised for each user and associated in the authentication database along with identifier and password). The reason for the nonce is that it is necessary to ensure that the use of the password is *fresh*. Even if a password is encrypted, an eavesdropper could capture it and submit it again later. The nonce is logically a timestamp, though it may in practice be generated e.g. as a random number, so the facility controller can be sure it has not been seen before without having to rely on a timer or, more importantly, allow only one user into the facility at any one time.

Though quite complex, this protocol has the defect that operation is entirely one way. The facility has only to check whether what the user offers matches its records and encoding conditions. Since the records might have been subverted, the protocol has its limits (quite apart from whether the encryption method used can be broken).

Aadditional security can be achieved by seeing whether a purported user can respond to a challenge from the controller.

Protocol 2

> In this case the user initially does no more than invoke logon (ring the bell) at the entry point. The system then sends their own nonce, e.g. random, string to the user. The user, who is supplied with a suitable offline coding device, enters the nonce string along with their own PIN into this device. The device combines the two, encrypts the result by applying a stored key, and displays the resulting number to the user who enters it into the system as his password and submits it to the facility controller. The facility controller, armed with knowledge of the user's PIN and the encryption key, repeats the computation and checks that she gets the same result.

This protocol is quite different from the previous one and seems suprisingly cumbersome. But it has important advantages. There are no stored passwords, though there are stored encryption keys at either end and a PIN record for the user at the system end. The password is guaranteed fresh for each system access, because it is newly generated (and can also be more secure than the average human-chosen password). Encryption plays a more central role since it is used to generate, and not merely encode, the password.

The weakness of the protocol is that it relies on the offline device which, in the ordinary world, could be stolen and, since it is not difficult to hypothesise PINs, could allow improper facility access. It is however a respectable protocol for use in establishments, like banks, which have separate personnel security.

Moreover this protocol, when viewed as an abstract protocol model, is limited in the same way as Protocol 1. It is based solely on determining whether one party, the user, is legitimate. It does nothing to satisfy a user's need to establish that the system is the one they think it is (e.g. is not a bogus ATM). More generally, authentication has to be allowed to be a peer-to-peer relationship with mutual authentication. This adds further complexity because it is important to maintain parity of information and control.

Protocol 3

> Here we have two users, conventionally Alice and Bob, and an intervening system controller Sam. Suppose Alice wishes to communicate with Bob. She signals Sam, using for this the personal key she has for interacting with Sam. Bob naturally also has a different personal key for communicating with Sam. In order to make Alice and Bob's direct communication secure, Sam creates a new session key for their interaction, and packages this up along with their two identifiers and a timestamp or nonce to indicate freshness. Sam also encrypts the package both for Alice with Alice's key, and for Bob with Bob's key. He then sends Alice her package, and also sends her Bob's package, for her to use as a letter of introduction to Bob

(she of course can't read it herself). When Alice now contacts Bob and presents the letter, Bob is able to establish that the letter is acceptable because it is encrypted with the key he shares with Sam, and that it is fresh, so he can respond to Alice. It may be appropriate for him to add a challenge step, as with Protocol 2, before doing this, to ensure that it is really Alice who is trying to communicate with him. However once communication has been successfully opened up, Alice and Bob can use their joint session key to send secure messages to one another.

Even this protocol, with what looks like rather careful detail, in fact has weaknesses. Thus for example it is vulnerable to anyone who can steal Alice's key for communicating with Sam, since this would enable the impersonator to establish session keys for communicating with Bob or other users.

The protocol essentially assumes that the users are members of a mutually trusting community and are primarily concerned that their communications with one another be protected against outside intrusion. Unfortunately members of communities are not always trustworthy. There are better forms of the protocol e.g. Kerberos, but these in turn have their own weaknesses.

These three examples are about authenticating human users. But it is clear with Protocol 3, that the agents can perfectly well be programs, and authenticating systems to one another is as critical as authenticating humans. Even when humans are involved, moreover, authenticating the individual may be less important than authenticating their host organisation, e.g. what bank is signalling may matter more than which of their clerks is bank representative. Finally, authentication applies to the content of messages as much as to who sends it. It is not much use to Alice to know that it is indeed Bob at the other end if she has no meens of knowing whether the actual message he is sending is authentic or has been interfered with en route.

The take-home lesson about authentication is thus: analyse your protocol design and implementation for *every* assumption and check these are reasonable in the circumstances in which the protocol is to be used.

Non-repudiation

In communication between parties, authentication is knowing where a signal came from and maintaining confidentiality is knowing where it went. Non-repudiation is acknowledging that the signal happened. But this is not just a matter of system evidence, e.g. in a log file. Ultimately there has to be a human human judge of whether, on the basis of all the evidence, the transaction has occurred.

Encryption

Crytography is a large and extremely technical subject, invoking heavy mathematics. This is therefore not the place to go into it. For present purposes the important points about encryption are:

how easy is it to break a secret code cipher in the absence of a key ?
who has the key ?
where is it appropriate to apply encryption ?

We have already referred to an extremely simple cipher, letter substitition, with its table key. The key lists the plain-letter/cipher-letter pairs; enciphering the header or body of a message involves substituting the appropriate cipher letter for the given plain letter, and deciphering is the reverse. Clearly both sender and receiver of the message have to have a copy of the key.

Ciphers like this are far too easy to break without the key. Ciphers can be made more complex, and hence encryption more secure, by stream or block techniques: in the first the way a plain letter is treated depends on its position in the plaintext, in the second a block of several letters is replaced. Things can be made more challenging to the code-breaker by introducing randomness so e.g. it is not obvious which letters in an encrypted string belong to the real message and which are just noise, produced by choosing a letter as instructed by some random (strictly, pseudo-random) number generator. In general, encryption is made more effective by making keys more opaque e.g. very long, by elaborating on the enciphering process, e.g. applying several cycles, and by smothering everything in huge numbers so, e.g. finding what numbers might have been multiplied together to produce them is a big computing deal. However, as machines become more powerful, once-large numbers become smaller. Moreover using random numbers in encrypting has a cost in requiring a subsidiary 'key', namely information about that the generation algorithm is and what 'seed' has been used to start it.

Quite apart from these issues, for any user wanting to communicate with many other users (or many facilities), it is a nuisance to have to have separate keys for each one. However it is clearly dangerous to assume that when a single key is known to a large number of people or kept in many places, it will be secure.

Public key encryption

Having a pair of keys for each user (facility) and making one of them public is a way of dealing with this problem. In public key encryption every agent has two keys, a public one and a private one. Anyone who wants to communicate with Alice can use her public key to encrypt their message, which can only be decrypted by Alice herself using her private key. Thus if Bob sends Alice a message encrypted with her public key it is secure because he alone knows what the message was before it was encrypted, and Alice is the only person who can get at it after it has been encrypted. There is no need, in particular, for anyone but Alice to know her decryption key.

Digital signatures

Digital signatures work the other way round: the signature can only be written (created) by one person, but can be read (verified) by anyone.

Of course, as with authentication, encryption applies as much to messages between programs and systems as between humans. For example one program may send another a message consisting of a password to be checked (as with an ATM and a bank), and commercial data may be encrypted for transmission between offices in different countries.

In a general way, it might appear that the more encryption the better. The examples in Part 1 illustrate many points for where it can be applied within a single system. However encryption has costs associated both with processing and key data conservation, and it is not necessary to encrypt every bit, i.e. 0 or 1, in a complete communication. What matters is knowing which bits to encrypt e.g. that it may be important to encrypt authentication exchanges to avoid letting undesirable users or programs into a system, but not necessarily the messages that are subsequently exchanged which could contain only routine information; or it could be appropriate to encrypt the messages as well, for extra security. There are encryption standards, and reputable encryption algorithms e.g. RSA, the DES algorithm, as well as protocols for establishing and using keys, including certifying the link between users and keys. What matters with encryption is getting sound expert advice on whether a proposed encryption regime is adequate for its intended system use.

Digital rights management

Security can be viewed primarily as protecting computer users and owners against other people. Encryption is also used in digital rights management to protect others (the holders of digital property, e.g. a music soundtrack) against computer owners who may want to misappropriate it, say by copying it. There are important issues here (though outside my present scope) about aggressive uses of security technology e.g. to protect monopoly commercial positions.

System structure and operation

Modern computer systems are very large, i.e. very powerful, and very complicated, for example because they consist of many machines. It is particularly important to recognise that while an individual may think of their PC as an independent machine that just happens to have a connection to the Internet, as a device for sending emails to others out there, and of the Web as a convenient remote encyclopedia, in reality pretty well everything is connected with everything else. This is why viruses spread so quickly.

The implications of this global connectivity for anyone concerned with security are:

first, that in general, any individual component agent (person, machine) will have multiple connections with others, i.e. be linked for one or more functions (filing, archiving, emailing, web browsing etc) with many other systems, implying many interfaces for which security has to be considered; and

second, that any agent will be connected with others indirectly as well as directly, quite possibly through many intervening systems, so security chains or dependencies have to be taken into account.

As this suggests, local security for any one component agent or system, in handling both what it receives and what it sends, has to be thoroughly organised. For instance, using an office analogy, if ordinary mail received through the post is put in pigeonholes for addressees to pick up themselves, the mailroom has to be careful to keep incoming FedExes separate as these have to be explicitly signed for. Equally, it is a mistake to put outgoing FedExes in the outgoing postbag as this may introduce a delay - discovery by the PostOffice, return to sender, restart with FedEx collector - which could seriously affect the status of the enclosed message.

But equally, security propagation between one agent or system and another at a distance has to be thoroughly understood. There is no point in my labelling a sealed envelope addressed to Dr X. and containing a personal reference, as Confidential if I cannot be sure that the envelope, still sealed, will reach Dr X's desk, for him alone to open. My initial security mechanism - sealing and labelling - is subverted if the envelope is opened in a secretary's office and the actual reference letter is left exposed, for anyone in the office to read, on a tray for Dr X to pick up when he comes in.

In both these situations, even the local one, the agent originating the action is not, and cannot be, in control of what follows, and has to rely on others to support and maintain their security requirement.

In general, that is, security in any individual case - i.e. for an action like sending an email, depositing a data record, reading a file entry, checking a logon, or for a state like that of an archive, a document being edited, or a calculation in progress, - depends on security in a system as a whole. This does not imply that the same security regime has to apply everywhere, just as it does not imply that any agent (human or machine) can only have a single role with particular security status. Allowing for different regimes for different purposes, and several different roles, provides the flexibility for systems to operate effectively. (For example, requiring all mail to be FedExed, and hence requiring everyone involved to have the role of FedEx sender/receiver or clerk, would be costly and a waste of everyone's time on the logistics of packing, logging, signing etc.)

At the same time, much of what is required for security is required for proper software engineering from the most mundane and basic point of view: no-one wants a computer operating system not to check what databases it is writing to, or not to worry about what happens when it tries to make modifications requested by two distinct agents to the same database at the same time, or not to bother when it sends the same email by multiple routes to increase the chances of at least one getting to its destination, and fast, with what happens to any later arrivals (a stockbroker's client does not necessarily want the broker to sell or buy more than one batch of shares).

The intimate relationship between security engineering and software engineering implies that decent software engineering should provide some of the necessary underpinning for security engineering in a narrower sense. But since software engineering in practice is often far from satisfactory and, further, specific security requirements may be more stringent than the default general ones, it is necessary to examine the requirements a security regime imposes and to check whether the system's foundations are actually adequate for the proposed security building or will simply let it slowly sink into the ground, or crack open and collapse.

One way of doing this is to assess a system's ability to maintain a security status hierarchy for agents (or more strictly, agent roles), that controls information flow and action scope. For example, anything an ordinary user can see can be seen by some privileged administrative users, and anything these can see can be seen by some top-level superuser, but the reverse does not apply: information flows in one direction only. In the most straightforward case, action could apply in the opposite direction: anything done by a bottom-level user can be cancelled by an administrative user, who in turn may be subjected to action by the superuser: more generally an action at the top would apply downward all the way. This is a very simple illustration, and an actual situation could be more complex: for example only some superuser actions might apply downwards, or some end-user information be readable upwards. The same notions of course apply to software as opposed to human agents. The general model of information flow referred to as the Bell-Lapadula model was developed for military contexts, but the flow idea is the kind of tool that can be usefully applied to analyse security, especially in terms of roles, in other contexts. Similar strategies, applying notions of chinese walls or boxes, can be used to check the security insulation of one part of a system from another.

But there is a real problem in computer security with *leakage*: somehow, someway, however things started with tight, or appropriate, security, the combination of system spread and operational speed means that security will never be more than a goal, not a fact. Good system design means minimising the consequences of leakage. It is especially important to recognise the dangers that 'bought-in' packages involve. These may be presented as complete (business) solutions to some functional requirement, but should never be accepted as plugin black boxes. They need detailed security analysis like any other software, in particular to establish that they do exactly what is specified and not something subtly, but damagingly, different.

Viruses, worms and denial of service

Though the concern of this note is with guarding systems against misuse by their users, whether careless or corrupt or elitist, rather than against invasion by teenage hackers, it is important to recognise that invasions can affect legitimate operations e.g. through changes of data or modifications of control regimes, even if they do not disrupt them entirely. The kinds of attacks represented by viruses and worms (the strictly technical differences between these do not matter here) can be far more damaging, and so can denial of service attacks where systems are prevented from working by e.g. tsunamis of email. Shutting proper users out of systems can lead to awkward, costly, or life-threatening situations. Many such attacks are extremely undiscriminating and the fact that some system 'ought' not to attract offensive attacks does not mean that it will not in practice suffer, and security engineering has to allow for this.

Files and databases

Security for files and databases looms large in any computing application. The points made about system security in the previous section are directly relevant to files and databases. However the fact that databases may contain large amounts of sensitive information about, or relating to, people or organisations, may be continually changing, may have very long life times, and may often also necessarily have many users, means that database security is particularly significant in practice.

Thus as well as guarding against accidental damage (whether to their content or their physical storage devices), and ensuring that database functions including 'delete item' are correctly and fully implemented, it is particularly important to provide proper control on (human or machine) agent access, and to ensure that the functions these agents can execute are clearly specified. Agent roles and their capabilities, and role-based access control, are valuable security mechanisms for this purpose, e.g. to distinguish powers to read only or to write as well. (Applying these mechanisms is in turn facilitated by convenient ways of modelling agents, e.g. as abstract 'objects', as in CORBA.)

Database security has many ramifications, e.g. keeping agent authorisations and role assignments up to date, ensuring that the archiving that is essential for significant databases is securely done and that the archives themselves are secure; ensuring that distributed databases, or databases that exist in multiple copies, are equally secure in all their parts or copies, and so forth.

In addition or rather, crucially, the social and economic functions that databases serve means that the definition of the setup (i.e. human as well as computational context) within which the database exists and is used is comprehensive enough to ensure that it captures management roles with responsibility, and accountability, for the database. This has be done as grounding for the other role assignments, even if the person(s) with the roles for which responsibility and accountability are specified never actually see or use the data themselves, and never go near the computer systems on which the databases live. Further, it is necessary, since databases have non-trivial lives, to address the question of responsibility and accountability over time, with their corresponding obligation to maintain database information for as long as it may affect its subjects. It is essential not to treat the way computational databases are implemented as a purely technical matter that is independent of, and hence of no concern in detail to, those who are actually responsible and accountable for what the databases hold and what they are used for.

Iris recognition - a cautionary example

Automated iris recognition appears to be an unrivalled means of achieving access security for individuals: since the human iris is effectively unique, it is possible to authenticate an individual, and hence allow them to access a store or facility. Further, iris recognition is a computational security device becase of the processing required to transform the input image of the iris and check it against the reference base of iris image transformations for people.

However the security is not as complete as might be supposed, and belief in the power of iris recognition security comes from considering it only from one point of view, namely that it is impossible to substitute an alternative eye. Thus if the file form is for X's eye, Y's eye can not be offered the input scanner and expected to match the file form. Thus Y cannot gain access where only X is allowed.

But since there are two parts to the device, the input eye image and the file form, it is also necessary to consider whether it is possible to substitute Y's image form in the file, thus allowing Y to impersonate X. Doing this depends on being able both to invoke the input image transformation software, and to deposit the resulting form in the file. It might also be impossible to sustain impersonation for long, for example if X appeared, was denied access, and succeeded in raising doubts about the status of the file version. However the *possibility* of tampering successfully with the file shows that iris recognition does not offer the total access security many suppose.

To improve security it would be desirable e.g. to keep the transformation software well separated from the file, ansd to supplement the use of iris recognition by other means of personal identification/authentication. The latter would unfortunately reduce the beautiful simplicity of the pure iris model; and it would not ensure that the file could never be violated. But it might be better than relying on iris recognition alone.

Take home message

Computer technology, in radically changing the way information is handled, means that **every** *aspect of security for that information has to be rethought. But it also means that, though computing offers new security devices, these have their own vulnerabilities which may be more difficult, because of system scale and complexity, to detect and remove, and which therefore leave information systems open to the misuse or damage that go along with having human beings anywhere around. These dangers are compounded by the fact that systems themselves are continually mod-*

ified in detail, often for good, independent reasons, and much more significantly by the fact that the environments in which systems operate evolve, so that the security assumptions and policies on which they are based no longer apply, and unforeseen security disaster can strike.

I am grateful to Roger Needham and Ross Anderson for comment, to them and the latter's book for my security education, and to John Naughton for further comment.

Ross Anderson, *Security Engineering*, Wiley, 2001.