Chapter 1

Introduction

The History of Herodotus

By Herodotus

> *For Histiæus, when he was anxious to give Aristagoras orders to revolt, could find but one safe way, as the roads were guarded, of making his wishes known; which was by taking the trustiest of his slaves, shaving all the hair from off his head, and then pricking letters upon the skin, and waiting till the hair grew again. Thus accordingly he did; and as soon as ever the hair was grown, he despatched the man to Miletus, giving him no other message than this- "When thou art come to Miletus, bid Aristagoras shave thy head, and look thereon." Now the marks on the head, as I have already mentioned, were a command to revolt...* [HerBC]

There are many aspects to "computer security", but they all derive from the study of security in general. In most cases, the same security problems that occur in society occur in one form or another in computers. For example, confidentiality problems result in concerns about locks, and encoding. Integrity problems result in concerns about signatures, and handshakes. In each of these, we can see simple examples from society, and the computer-related versions follow the same lines (only a million times faster).

Histiæus ensured confidentiality by hiding his message in such a way that it was not immediately apparent, a technique so clever that it was used again by German spies in the 1914-1918 war. This sort of information-hiding is now called steganography, and is the subject of active research.

Cæsar is reputed to have encoded messages to his battalions, a technique now called cryptography, and the use of agreed protocols between armies to ensure the correct conduct of a war is seen over and over again in history. You will notice that we have begun with examples taken from the world of warfare, and throughout this course, you will find references to military conduct, procedures and protocols. This should not be a surprise to you given the nature of *security*.

1.1 Range of topics

The study of computer security can cover a wide range of topics, and for this introductory course, I have decided to concentrate on the following distinct subject areas:

- **Mathematical, physical and legal preliminaries.** Some aspects of computer security require an appreciation for various mathematical, physical and legal laws. We will review the principal ones.
- **Security models.** These models provide formal (read *mathematical*) ways of looking at computer security in an abstract manner. Consider the situation that you adopt a formal security model and the model is provably secure. If you then ensure that all components of your system comply with the model, you can be sure of the security of your system.
- **Secrecy.** Much of modern-day commerce relies on secure transfer of information, and this security relies on exchange of secret keys. In addition, we often just want things to be secret, and encrypt documents to ensure this. The expanding global digital world shrinks the distance between you and an attacker. A few years ago, you could just have locks on the doors, and not invite criminals home for dinner, but now criminals have an electronic access point into your living room, your bank and so on. We will investigate secrecy techniques.
- **Insecurity.** Most computer systems are dangerously easy to subvert, and it is a scary world out there! Apart from an adversary gaining some level of control over your system, consider the insecurity you might feel when you sign a contract, and then the other party doesn't. Sometimes our concern is not with secrecy, but with subtleties like non-repudiation. We will investigate some common hacking and insecurity strategies, and examine some techniques for reducing risk to your systems.
- **Safety/control software and hardware.** Operating systems and distributed systems are complex entities, and various techniques for improving the security of such systems will be examined.
- **Assurance.** How can we convince ourselves (or our employer) that the computer system is to be trusted? Building assurance is best done by adopting formal methods to confirm, specify and verify the behaviour of systems.
- **Protocols.** Some aspects of security are determined by the way in which we do things (the protocol), rather than what is actually done.

1.2 Mathematical, physical and legal preliminaries

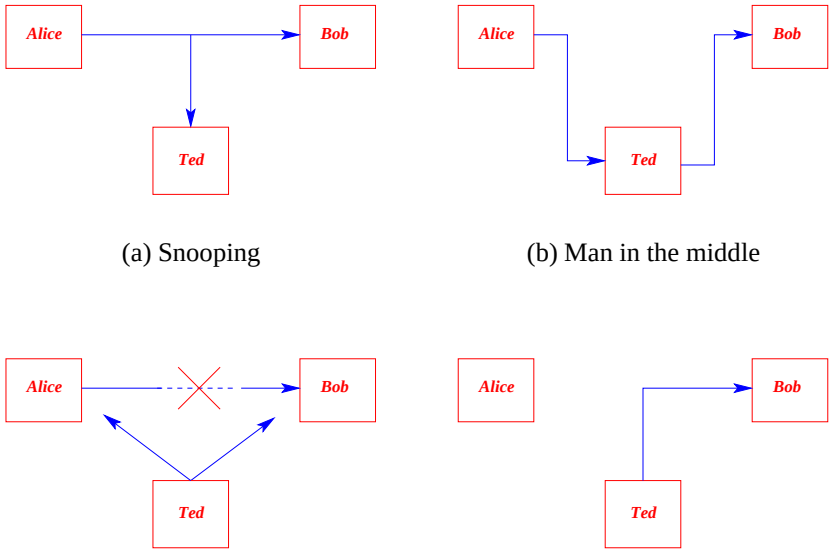
We begin with brief lists/descriptions of some of the underlying aspects of security *services*, *threats* and *attacks*. We distinguish between security *policies* and *mechanisms*, and investigate

relevant *mathematical* and *physical* laws. We will briefly explore some *legal* aspects. Please read more detail from the textbook.

Our first list classifies three aspects of security *services*:

- **confidentiality**: concealment of information or resources;
- **integrity**: trustworthiness of data or resources;
- **availability**: preventing denial-of-service.

In Figure 1.1 are some diagrammatic representations of examples of threats to computer systems.



(a) Snooping (b) Man in the middle

(c) Denial of service (d) Spoofi ng

Figure 1.1: Examples of threats to systems

Our list gives us an attempt to classify these *threats* to a system. Note that lists such as these vary from textbook to textbook:

- **disclosure**: unauthorized access (snooping);
- **deception**: acceptance of false data (man-in-the-middle);
- **disruption**: prevention of correct operation (denial-of-service);
- **usurpation**: unauthorized control (spoofing).

We differentiate between a security *policy* and a security *mechanism*:

- **policy**: what is allowed/disallowed;
- **mechanism**: ways of enforcing a policy

For example, at NUS, we have an IT policy which includes a range of clauses regarding security concerns, such as:

> ***4.2 Undermining System Integrity***
>
> *Users must not undermine the security of the IT Resources, for example, by cracking passwords or to modify or attempt to modify the files of other Users or software components of the IT Resources.*

This policy is enforced by various software tools. If you look at the NUS IT policy document[1] which you have all signed, you will notice the following clause:

> ***6.3 Use Of Security Scanning Systems***
>
> *Users consent to the University's use of scanning programs for security purposes at system level for computers and systems that are connected to the University's network. This is to ensure that any computers or systems attached to the network will not become a launching pad for security attack and jeopardise the IT Resources. System level scanning includes scanning for security vulnerabilities and virus detection on email attachments. Users' files and data are excluded from the scanning.*

In addition to the preceding concept preliminaries, we will also be considering fundamental *mathematical* and *physical* laws and procedures. In [Wag], Wagner introduces mathematical notions of interest, including a range of operators and algorithms that should be known by anybody interested in cryptography and computer security.

A large amount of modern computer security is concerned with ensuring the confidentiality, integrity and availability of computer *communication* systems. We will be extending the mathematical notions with introductory physical ones of communication, randomness and entropy, each of which has relevance to the study of communication systems.

[1] Found at https://online.nus.edu.sg/DB/infoboard/27781.doc.

1.3 Security models

The term security model refers to a range of formal policies for specifying the security of a system in terms of a (mathematical) model. There are various ways of specifying such a model, each with their own advantages and disadvantages. The following is an incomplete list:

- The Bell-LaPadula [BL75] model (no read-up, no write-down) provides a military viewpoint to assure *confidentiality* services. There is a brief introduction to this which is worth reading in [MP97].
- The Biba [Bib75] and Clark-Wilson [CW87] models attempt to model the trustworthiness of data and programs, providing assurance for *integrity* services.

Having a model of course is not the end of the story. We need to be able to determine properties of the model, and to verify that our implementations of the security model are valid. However the above models have formed the basis of various trusted operating systems, and later in the course we will examine this in more detail.

Modelling for availability is a little more problematic, as the range of threats is wide, and the possibility of prevention of all such threats is very small.

1.4 Secrecy

Lets look at three time periods...

1.4.1 2000 years ago...

Cæsar (100-44BC) is reputed to have encoded messages to his battalions. He is supposed[2] to have done this using (what is now called) a Cæsar cipher, in which we replace each Roman letter in a message, with another Roman letter, obtained by rotating the alphabet some number of characters:

I C L A V D I V S

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

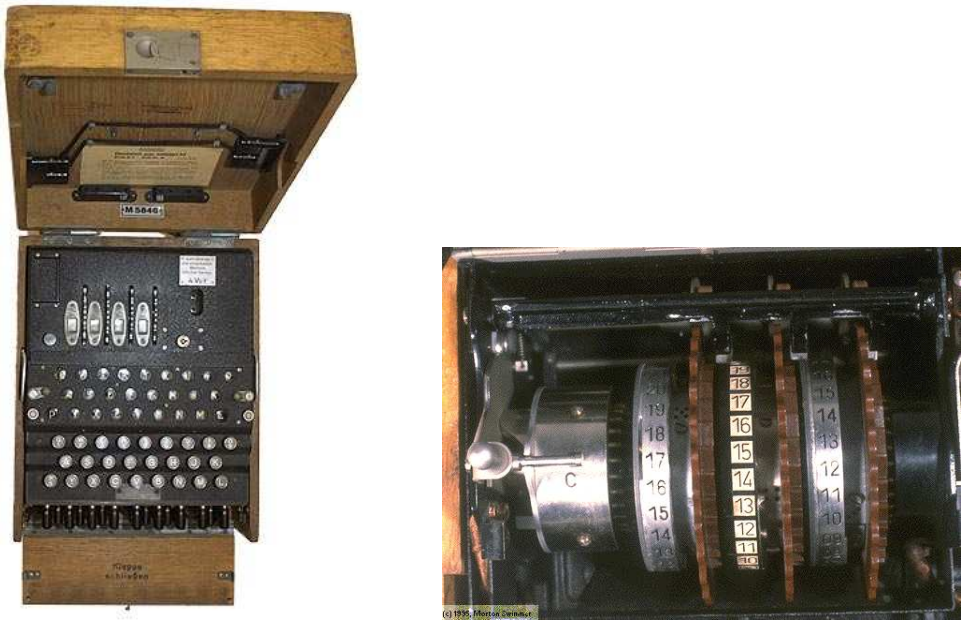
E Y G V Q Z E Q O

We can specify a Cæsar cipher by just noting the number of characters that the alphabet is rotated.

[2] Suetonius [SueAD] wrote "*There are also letters of his to Cicero, as well as to his intimates on private affairs, and in the latter, if he had anything confi dential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet, that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely D, for A, and so with the others*".

1.4.2 60 years ago...

In the 1920's, a German company marketed a series of devices for encrypting and decrypting messages. The devices used electro-mechanical rotors to generate what appeared to be random characters from a source message. However a matching device, with a special key, could decode the messages. These *Enigma* devices were extensively used by the German military to communicate before and during World War II, in the belief that the "random" characters could not be decoded.



(a) Top view (b) The rotors

Figure 1.2: The Enigma machine

If you believe in the movie world (as of course every right-thinking person does), then you may already know that the American Navy captured a German submarine and recovered an Enigma machine and codebook, leading to the allies being able to decode the German military messages (U-571). The movie is of course nonsense, but the real story is just as exciting, and does not require the heroes to be good looking and have easily pronouncable names.

> *The first attempts to break the machines were made in Poland in 1928, after the Poles intercepted a machine en-route to the German embassy in Warsaw. Three students from the Department of Mathematics at the University of Poznan were assigned to work on the problem. They were able to decode some messages from the first simple machine, but the German army were using a more secure machine, with an extra*

level of encoding. In a collaboration with French spies, information about the extra encoding was uncovered, and by 1933, the Polish Ciphers Office was able to decode messages, although slowly. From then until the invasion of Poland in September 1939, the Poles were able to decipher over 100,000 transmissions.

In 1938, the German's changed the encoding of the Enigma machines, and the Poles had to develop a machine to decode the new messages. This machine was completed quickly, and the Poles were aware of the date of the imminent invasion of Poland. As a result of this information, Poland delivered working Enigma copies to the English about two weeks before the invasion of Poland. The English were able to decode German low-security messages from August 1939.

English cryptographers at Bletchley Park, including Alan Turing, developed many systems for decoding encoded German (and other) transmissions. On 9 May 1941 the Royal Navy escort destroyers Bulldog and Broadway captured the U-110 submarine, complete with a genuine Enigma machine, and active code books. From this date on, the English could decode most German military transmissions.

The Polish systems and ideas helped the English develop a hardware system to decode Enigma keys very quickly (they changed daily). These machines are considered one of the precursors to modern-day computers, but were considered state secrets until 1990.

1.4.3 Today...sssshhhh

From the manual pages for ssh, the secure-shell:

The program ssh is for logging into a remote machine and for executing commands on a remote machine. It provides secure encrypted communications between two untrusted hosts over an insecure network. Other connections can also be forwarded over the secure channel. Users must prove their identity to the remote machine using one of several methods depending on the protocol version.

One method is the rhosts or hosts.equiv method combined with RSA-based host authentication. If the server can verify the client's host key, only then login is permitted. This authentication method closes security holes due to IP spoofing, DNS spoofing and routing spoofing.

The scheme is based on public-key cryptography: cryptosystems where encryption and decryption are done using separate keys, and it is not possible to derive the decryption key from the encryption key. RSA is one such system.

Schemes such as secure-shell are typical of modern computer secrecy systems. They rely on encodings that are believed to be difficult to decode, and protocols of message exchange that are believed to be secure.

1.5 Insecurity

Insecurity begins with closed doors, locks, and the realization that certain people can pick locks. Consider a locked air-conditioned room containing a file server. How secure is this? Well...

- The lock can be picked, or the door kicked in.
- The console of the server computer may be password protected, but it may be rebooted with a different disk.
- The reboot process may be (BIOS) password protected, but the case of the computer may be opened and the disk removed.
- And so on...

Well, you argue, we would know afterwards because of the bootmarks on the door, the logfiles of the computer, the missing disk.

For a different type of insecurity consider the widespread use of computer screens. An interested person can record the content displayed on any computer screen, and do so from a distance, with no interconnecting wires. In van Eck's famous paper [vE85] he describes driving around London, and viewing computer screens in nearby buildings. The equipment used in this study only cost about $15, but even so, van Eck suggests that it would be possible to monitor screens at a distance of 1km. A determined adversary with a large budget should be able to do better. This form of spying has been known of for at least 40 years. You may be interested to read a more recent paper [KA98] recording various ways to subvert this sort of remote monitoring of VDU screens, including the use of specialized fonts.

Figure 1.3: Trinity en-route to kicking in a door

With the advent of widespread interconnectivity between computers, it can be relatively easy to kick in doors without even using your feet. In class we will see how the technique shown in Figure 1.3 was used to change the world.

In the world of e-commerce, we may be interested in less direct forms of deception, for example non-repudiation:

- the buyer cannot order an item and then deny the order took place;
- the seller cannot accept money or an order and then later deny that this took place.

Intrusive hacking is common on the Internet. There are groups of people who build farms of subservient machines, so they can later use them for various purposes:

> *At first, it looked as if some students at the Flint Hill School, a prep academy in Oakton, Va., had found a lucrative alternative to an after-school job. Late last year, technicians at America Online traced a new torrent of spam, or unsolicited e-mail advertisements, to the school's computer network.*
>
> *On further inquiry, though, AOL determined that the spammers were not enterprising students. Instead, a spam-flinging hacker who still has not been found had exploited a software vulnerability to use Flint Hill's computers to relay spam while hiding the e-mail's true origins.*
>
> *It was not an isolated incident. The remote hijacking of the Flint Hill computer system is but one example among hundreds of thousands of a nefarious technique that has become the most common way for spammers to send billions of junk e-mail messages coursing through the global Internet each day.* [Han03]

We are familiar by now with computer virusses, especially the boot-sector virusses which hide their code in the boot sector of a disk. One of the earliest widely distributed virusses was the stoned virus for DOS, written by a student from New Zealand. A virus contains code that replicates, attaching itself to a program, boot sector or document. Some viruses do damage as well.

By contrast, a worm is a program that makes copies of itself, transferring from one disk drive to another, or one machine to another. Perhaps the most famous worm was the Morris worm in 1988:

> *On the evening of 2 November 1988, someone infected the Internet with a worm program. That program exploited flaws in utility programs in systems based on BSD-derived versions of UNIX. The flaws allowed the program to break into those machines and copy itself, thus infecting those systems.*
>
> *This program eventually spread to thousands of machines, and disrupted normal activities and Internet connectivity for many days.* [Spa88].

The author of the worm, Robert Morris, was convicted and fined $10,050 in 1990, and is currently a professor in the Parallel and Distributed Operating Systems group at MIT, lecturing in distributed systems areas.

1.6 Safety/control software

A naive approach to security might involve attempting to ensure that all programs that run on a computer are *safe*, and that all users of computer systems are *trustworthy*. This approach is not immediately practical, as there are many programs, and checking even one program is a non-trivial task. The computer *operating system* normally provides some level of software and hardware security for computer systems, combined with some level of user authorization.

User authorization means passwords! Modern multi-user operating systems have some level of password protection as you are all aware, and these systems have grown in complexity over the years. The historical article [MT79] shows the changes in the UNIX security password mechanism is the early years of UNIX development. Note that before UNIX systems programmers started working on the problem of password security, the general technique was to put the (unencrypted) passwords in a file that was difficult to read and write. Once this file was compromised, then the whole system was compromised.

Hardware security in operating systems has been studied in CS2106 (Operating Systems) and other courses. The Kernel/Supervisor bit, processor ring0, memory protection/mapping hardware and so on are all examples of hardware security systems intended to co-operate with the OS to enhance system security.

Software security in operating systems takes many forms. The forms range from ad-hoc changes to operating systems to fix security loopholes as they are found, through to operating systems built from the ground up to be secure.

TCP wrappers are one technique involving a change to systems to fix possible security loopholes. Many attacks to UNIX systems came through poorly controlled TCP or UDP ports. The TCP wrapper protects all such ports, providing a single point of control for access to each port. The default installation of TCP wrappers disables *all* port access, which you then re-enable on a case-by-case basis.

As an example of an OS built to be secure, those wonderful people at NSA have incorporated the results of several research projects in a security-enhanced Linux system:

> *This version of Linux has a strong, flexible mandatory access control architecture incorporated into the major subsystems of the kernel. The system provides a mechanism to enforce the separation of information based on confidentiality and integrity requirements.*
>
> *This allows threats of tampering and bypassing of application security mechanisms to be addressed and enables the confinement of damage that can be caused by malicious or flawed applications.* [PLF98]

You can read about SELinux at

http://www.nsa.gov/selinux/index.html

The Java virtual machine has a built-in security model [GMPS97], as it was built from the ground up with security concerns paramount.

Microsoft have their own view of the security of NT versus the Linux operating system, which is viewed as a threat to Microsoft because of it's cost and stability[3]:

> ***Myth:*** *Linux is more secure than Windows NT*
>
> ***Reality:*** *Linux security model is weak*
>
> *All systems are vulnerable to security issues, however it's important to note that Linux uses the same security model as the original UNIX implementations: a model that was not designed from the ground up to be secure.*
>
> *Linux only provides access controls for files and directories. In contrast, every object in Windows NT, from files to operating system data structures, has an access control list and its use can be regulated as appropriate. Linux security is all-or-nothing. Administrators cannot delegate administrative privileges: a user who needs any administrative capability must be made a full administrator, which compromises best security practices. In contrast, Windows NT allows an administrator to delegate privileges at an exceptionally fine-grained level. Linux has not supported key security accreditation standards. Every member of the Windows NT family since Windows NT 3.5 has been evaluated at either a C2 level under the U.S. Government's evaluation process or at a C2-equivalent level under the British Government's ITSEC process. In contrast, no Linux products are listed on the U.S. Government's evaluated product list.*

1.7 Assurance

The term assurance is related to trust. By careful and formal specification, design and implementation we can increase our assurance related to a computer system. The ITSEC process involves detailed examination and testing of the security features of a system.

> *During the 1980s, the United Kingdom, Germany, France and the Netherlands produced versions of their own national criteria. These were harmonised and published as the Information Technology Security Evaluation Criteria (ITSEC). The current issue, Version 1.2, was published by the European Commission in June 1991. In September 1993, it was followed by the IT Security Evaluation Manual (ITSEM) which specifies the methodology to be followed when carrying out ITSEC evaluations.*

[3] Please note that this comes directly from a Microsoft web site, and may have a fair amount of hype. It does not mention for example that the C2 security classification for NT is only for when the system has no network connections.

> *The Common Criteria represents the outcome of international efforts to align and develop the existing European and North American criteria. The Common Criteria project harmonises ITSEC, CTCPEC (Canadian Criteria) and US Federal Criteria (FC) into the Common Criteria for Information Technology Security Evaluation (CC) for use in evaluating products and systems and for stating security requirements in a standardised way. Increasingly it is replacing national and regional criteria with a worldwide set accepted by the International Standards Organisation (ISO15408).*

In [Woo98], elements of the first certification of a smart-card system under the European ITSEC level 6 certification are outlined. This process involved verification of the specification with independent systems, and a formal process for the implementation, deriving it from the specification using the *refinement* process.

1.8 Protocols

Sometimes the protocol we follow can be crucial to the security of a system. Consider the communications system shown in Figure 1.4.

Figure 1.4: A secure communication system

By following a specific protocol we can use it to transfer documents securely.

1.9 Term definitions

Here are some term definitions gleaned from various on-line technology dictionaries:

- virus: an unwanted program which places itself into other programs, which are shared among computer systems, and replicates itself.
- worm: an independent program that replicates from machine to machine across network connections, often clogging networks and computer systems as it spreads.
- steganography: the hiding of a secret message within an ordinary message and the extraction of it at its destination.
- cryptography: the science of information security.
- cryptology: the mathematics, such as number theory, and the application of formulas and algorithms, that underpin cryptography and cryptanalysis.
- cryptanalysis: the study of ciphers, ciphertext, or cryptosystems (that is, to secret code systems) with a view to finding weaknesses in them that will permit retrieval of the plaintext from the ciphertext, without necessarily knowing the key or the algorithm. This is known as breaking the cipher, ciphertext, or cryptosystem.
- cipher: any method of encrypting text (concealing its readability and meaning).
- block cipher: one that breaks a message up into chunks and combines a key with each chunk.
- stream cipher: one that applies a key to each bit, one at a time.

1.10 Summary of topics

In this section, we introduced the following topics:

- An introduction to computer security
- Some definitions

Supplemental questions for chapter 1

1. What is an acrostic? Give an example of one.
2. Differentiate between a Cæsar cipher and a Vigenère cipher.
3. In his column "Why cannot?" [Per03] in Streats, June 19, 2003, Geoffrey Pereira was annoyed that he had to key ctrl-alt-del to bring up the password prompt. He discovered how to bypass this sequence to save time, and this will be replicated to the 800 or so other employees of his company. Geoffrey seemed to miss discovering a specific reason for this mode of operation, and by bypassing the key sequence, he opens his company to a particular kind of attack. Discover the *reason*, and the *attack*.
4. What do you think of the *integrity* of Histiaeus's solution to his messaging problem?

Further study

- Textbook Chapter 1
- Monitoring computer screens (van Eck [vE85])
  http://jya.com/emr.pdf
- Overcoming Tempest monitoring [KA98]
  http://www.cs.rice.edu/~dwallach/courses/comp527_s2000/ih98-tempest.pdf
- The Morris worm [Spa88]
  ftp://ftp.cs.purdue.edu/pub/reports/TR823.PS.Z
- Military mathematical modelling of security [MP97]
  http://80-ieeexplore.ieee.org.libproxy1.nus.edu.sg/xpl/tocresult.jsp?isNumber=13172

Bibliography

[Bib75] K. Biba. Integrity consideration for secure computer systems. Technical Report MTR-3153, MITRE Corporation, Bedford, MA, April 1975.

[BL75] D. Bell and L. LaPadula. Secure Computer System: Unified Exposition and Multics Interpretation. Technical Report MTR-2997, Rev. 1, MITRE Corporation, Bedford, MA, March 1975.

[CW87] D. Clark and D. Wilson. A Comparison of Commercial and Military Security Policies. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pages 184–194, April 1987.

[Den71] P.J. Denning. Third Generation Computer Systems. *Computing Surveys*, 3(4):175–216, Dec 1971.

[Den76] D.E. Denning. A Lattice Model of Secure Information Flow. *Comm. ACM*, 19(5):236–242, May 1976.

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[GMPS97] L. Gong, M. Mueller, H. Prafullchandra, and R. Schemers. Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2. In *USENIX Symposium on Internet Technologies and Systems*, pages 103–112, Monterey, CA, 1997.

[Han03] S. Hansell. Unsuspecting Computer Users Relay Spam. *New York Times*, 20th May 2003.

[HerBC] Herodotus. The History of Herodotus. 440 B.C.

[KA98] M.G. Kuhn and R.J. Anderson. Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations. *Lecture Notes in Computer Science*, 1525:124–142, 1998.

[MP97] D. Mackenzie and G. Pottinger. Mathematics, Technology, and Trust: Formal Verification, Computer Security, and the U.S. Military. *IEEE Annals of the History of Computing*, 19(3):41–59, 1997.

[MT79] R. Morris and K. Thompson. Password security: A case history. *CACM*, 22(11):594–597, 1979.

[Per03] G. Pereira. Why Cannot? *Streats*, 19th June 2003.

[PLF98] P.A. Muckelbauer R.C. Taylor S.J. Turner P.A. Loscocco, S.D. Smalley and J.F. Farrell. The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. In *21st National Information Systems Security Conference*, pages 303–314, October 1998.

[Sha48] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[Spa88] E.H. Spafford. The Internet Worm Program: An Analysis. Technical Report Purdue University CSD-TR-823, West Lafayette, IN 47907-2004, 1988.

[SueAD] Suetonius. De Vita Caesarum, Divus Iulius (The Lives of the Caesars, The Deified Julius). 110 A.D.

[vE85] W. van Eck. Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk. *Computers and Security*, 4:269–286, 1985.

[Wag] N. Wagner. The Laws of Cryptography with Java Code.

[Woo98] J. Woodcock. Industrial-strength Refinement. In J. Grundy, M. Schwenke, and T. Vickers, editors, *International Refinement Workshop and Formal Methods Pacific '98*, pages 33–44, 1998. Keynote talk.