CHAPTER

9

Banking and Bookkeeping

Computers are not (yet?) capable of being reasonable any more than is a Second Lieutenant.

—CASEY SCHAUFLER

Against stupidity, the Gods themselves contend in vain.

—J.C. FRIEDRICH VON SCHILLER

9.1 Introduction

Banking systems include the back-end bookkeeping systems that record customers' account details and transaction processing systems such as cash machine networks and high-value interbank money transfer systems that feed them with data. They are important for a number of reasons.

First, bookkeeping was for many years the main business of the computer industry, and banking was its most intensive area of application. Personal applications such as Netscape and Powerpoint might now run on more machines, but accounting is still the critical application for the average business. So the protection of bookkeeping systems is of great practical importance. It also gives us a well-understood model of protection in which confidentiality plays almost no role, but where the integrity of records (and their immutability once made) is of paramount importance.

Second, transaction processing systems—whether for small debits such as $50 cash machine withdrawals or multimillion-dollar wire transfers—were the applications that launched commercial cryptography. Banking applications drove the development not just of encryption algorithms and protocols, but also of the supporting technologies, such as tamper-resistant cryptographic processors. These processors provide an important and interesting example of a trusted computing base that is quite different from

the hardened operating systems discussed in the context of multilevel security. Many instructive mistakes were first made (or at least publicly documented) in the area of commercial cryptography. The problem of how to interface crypto with access control was studied by financial cryptographers before any others in the open research community.

Third, an understanding of basic electronic banking technology is a prerequisite for tackling the more advanced problems of electronic commerce in an intelligent way. In fact, many dot-coms fall down badly on basic bookkeeping, which is easy to overlook in the rush to raise money and build a Web site.

Finally, banking systems provide another example of multilateral security, but aimed at authenticity rather than confidentiality. A banking system should prevent customers from cheating each other or the bank; it should prevent bank employees from cheating the bank or its customers; and the evidence it provides should be sufficiently strong that none of these principals can get away with falsely accusing another principal of cheating.

9.1.1 The Origins of Bookkeeping

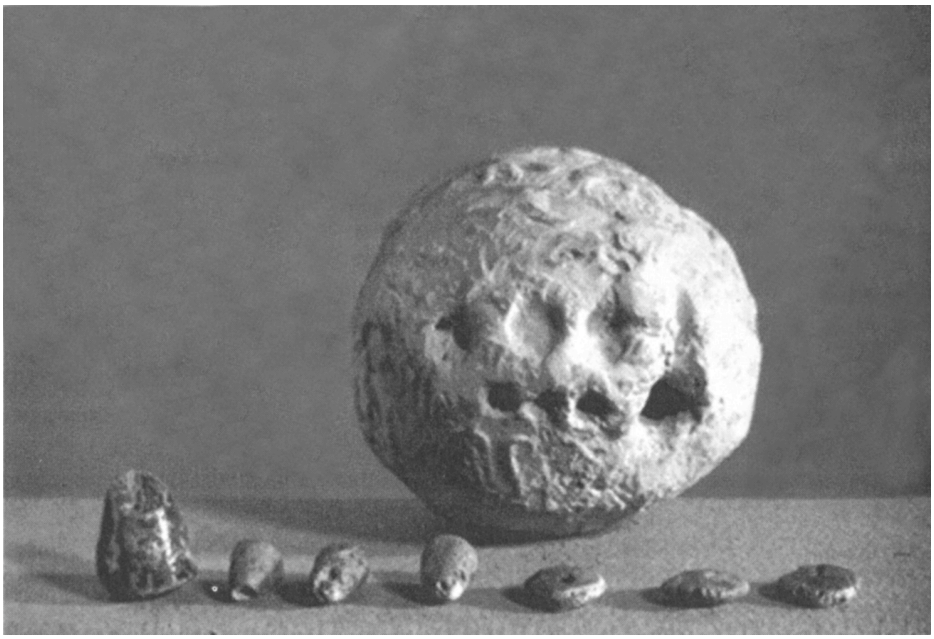

Figure 9.1 Clay envelope and its content of tokens from Susa, Iran, ca. 3300 BC (courtesy Denise Schmandt-Besserat and The Louvre Museum).

Bookkeeping appears to have started in the Neolithic Middle East in about 8500 BC, just after the invention of agriculture [678]. When people started to store and trade the food they had produced, they needed a way to keep track of which village member had put how much in the communal warehouse. To start with, each unit of food (sheep, wheat, oil, . . .) was represented by a clay token, or *bulla*, which was placed inside a

clay envelope and sealed by rolling it with the pattern of the warehouse keeper (see Figure 9.1). When the farmer wanted to get his food back, the seal was broken by the keeper in the presence of a witness. (This is may be the oldest known security protocol.) By about 3000 BC, this had led to the invention of writing [609]; after another thousand years, we find equivalents of promissory notes, bills of lading, and so on. At about the same time, metal ingots started to be used as an intermediate commodity, often sealed inside a bulla by an assayer. In 700 BC, Lydia's King Croesus started stamping the metal directly, and thus invented coins [625]; by the Athens of Pericles, there were a number of wealthy individuals in business as bankers [338].

The next significant innovation dates to the time of the Crusades. As the Dark Ages came to a close and trade started to spread, some businesses became too large for a single family to manage. The earliest of the recognizably modern banks date to this period; by having branches in a number of cities, they could finance trade efficiently. But as the economy grew, it was necessary to hire managers from outside, and the owner's family could not supervise them closely. This brought with it an increased risk of fraud, and the mechanism that evolved to control it was double-entry bookkeeping. This appears to have been invented sometime in the 1300s, though the first book on it did not appear until 1494, after the invention of the printing press [222].

9.1.2 Double-entry Bookkeeping

The idea behind double-entry bookkeeping is, like most hugely influential ideas, extremely simple. Each transaction is posted to two separate books, as a credit in one and a debit in the other. For example, when a firm is paid $100 by a debtor, the amount is entered as a debit in the accounts receivable book (the firm is now owed $100 less) and as a credit in the cash account book (the firm now has $100 more cash). At the end of the day, the books should *balance*, that is, add up to zero; the assets and the liabilities should be equal. (Any profit the firm has made is a liability to the shareholders.) In all but the smallest firms, the books will be kept by different clerks, and have to balance at the end of every month (at banks, every day). By suitable design of the ledger system, we can see to it that each shop, or branch, can be balanced separately. Thus most frauds will need the collusion of two or more members of staff; and this principle of *split responsibility*, also known as *dual control*, is complemented by audit.

Many computer systems are used for bookkeeping tasks, and implement variations on the double-entry theme. However, the control is often illusory. The double-entry features may be implemented only in the user interface, while the underlying file formats have no integrity controls. And even if the ledgers are kept on the same system, someone with root access—or with physical access and a debugging tool—may be able to change two or more records so that the balancing controls are bypassed. It may also be possible to evade the balancing controls in various ways; staff may notice bugs in the software and take advantage of them.

So how can we organize and formalize our protection goals?

9.2 How Bank Computer Systems Work

Banks were among the first large organizations to use computers for bookkeeping. They began to do so in the late 1950s and early 1960s, with applications such as check processing, and once they found that even the slow and expensive computers of that era were much cheaper than armies of clerks, they proceeded to automate most of the rest of their operations during the 1960s and 1970s.

A typical banking system has a number of data structures. There is an *account master file*, which contains each customer's current balance together with previous transactions for a period of perhaps 90 days; a number of *ledgers*, which track cash and other assets on their way through the system; various *journals*, which hold transactions that have been received from teller stations, cash machines, check sorters, and so on, but not yet entered in the ledgers; and an *audit trail* that records which staff member did what and when.

The processing software that acts on these data structures will include a suite of overnight batch-processing programs, which apply the transactions from the journals to the various ledgers and the account master file. The online processing will include a number of modules that post transactions to the relevant combinations of ledgers. For example, when a customer pays $100 into a savings account, the teller will make a transaction that records a debit to the savings account ledger of $100 (the bank now has an increased liability to the customer), while crediting the same amount to the ledger recording the amount of cash in the drawer. The fact that all the ledgers should always add up to zero provides an important check; if the bank (or one of its branches) is ever out of balance, an alarm will go off and people will start looking for the cause.

The invariant provided by the ledger system is checked daily during the overnight batch run; this means that a programmer who wants to add to his own account balance will have to take the money from some other account, rather than just create it out of thin air by tweaking the account master file. Just as in a traditional business one has different ledgers managed by different clerks, so in a banking data processing shop there are different programmers in charge of them. In addition, all code is subjected to scrutiny by an internal auditor and to testing by a separate test department. Once the code has been approved, it will be run on a production machine that does not have a development environment, but only approved object code and data.

9.2.1 The Clark-Wilson Security Policy Model

Although such systems have been in the field since the 1960s, a formal model of their security policy was only introduced in 1987, by David Clark and David Wilson (the former was a computer scientist, and the latter an accountant) [187]. In their model, some data items are constrained so that they can be acted on only by a certain set of transformation procedures.

More formally, there are special procedures whereby data can be input—turned from an *unconstrained data item*, or UDI, into a *constrained data item*, or CDI; *integrity verification procedures* (IVPs) to check the validity of any CDI (e.g., that the books balance); and *transformation procedures* (TPs), which may be thought of in the banking case as transactions that preserve balance. In the general formulation, they maintain the integrity of CDIs; they also write enough information to an append-only CDI (the

audit trail) for transactions to be reconstructed. Access control is by means of triples *(subject, TP, CDI)*, which are so structured that a shared control policy is enforced. In the formulation in Amoroso [15]:

1. The system will have an IVP for validating the integrity of any CDI.
2. The application of a TP to any CDI must maintain its integrity.
3. A CDI can only be changed by a TP.
4. Subjects can only initiate certain TPs on certain CDIs.
5. Triples must enforce an appropriate separation of duty policy on subjects.
6. Certain special TPs on UDIs can produce CDIs as output.
7. Each application of a TP must cause enough information to reconstruct it to be written to a special append-only CDI.
8. The system must authenticate subjects attempting to initiate a TP.
9. The system must let only special subjects (i.e., security officers) make changes to authorization-related lists.

A number of things bear saying about Clark-Wilson.

First, unlike Bell-LaPadula, Clark-Wilson involves maintaining state. Quite apart from the audit trail, this is usually necessary for dual control as you have to keep track of which transactions have been partially approved—such as those approved by only one manager when two are needed. If dual control is implemented using access control mechanisms, it typically means holding partially approved transactions in a special journal file. This then means that some of the user state is actually security state, which in turn makes the trusted computing base harder to define. If it is implemented using crypto instead, such as by having managers attach digital signatures to transactions of which they approve, there can be problems managing all the partially approved transactions so that they get to a second approver in time.

Second, the model doesn't do everything. It captures the idea that state transitions should preserve an invariant, such as balance, but not that state transitions should be correct. Incorrect transitions, such as paying into the wrong bank account, are still allowed.

Third, Clark-Wilson ducks the hardest question, namely: how do we control the risks from dishonest staff? Rule 5 says that "an appropriate separation of duty policy" must be supported, but nothing about what this means.

9.2.2 Separation of Duties

There are basically two kinds of separation of duty policy: dual control and functional separation.

In dual control, two or more different staff members must act together to authorize a transaction. The classic military example is in nuclear command systems, which may require two or three officers to turn their keys simultaneously in consoles that are too far apart for a single person to operate. (I discuss nuclear matters further in Chapter 11, "Nuclear Command and Control.") The classic civilian example is when a bank issues a letter of guarantee, which will typically undertake to carry the losses should a loan made by another bank go sour. If a single manager could issue such an instrument, then

an accomplice could plunder the guaranteed loan account at the other bank, and the alarm might not be raised for months. I discuss this further in Section 9.3.2.

With functional separation of duties, two or more different staff members act on a transaction at different points in its path. The classic example is corporate purchasing. A manager makes a purchase decision and tells the purchasing department; a clerk there writes a purchase order; the store clerk records the arrival of goods; an invoice arrives at accounts; the accounts clerk correlates it with the purchase order and the stores receipt, and cuts a check; the accounts manager signs the check.

However, it doesn't stop there. The manager now gets a debit on her monthly statement for that internal account; her boss reviews the accounts to make sure the division's profit targets are likely to be met; the internal audit department can descend at any time to audit the division's books; and when the external auditors come in once a year, they will check the books of a randomly selected sample of departments. Finally, when frauds are discovered, the company's lawyers may make vigorous efforts to get the money back.

The model can be described as *prevent-detect-recover*. The level of reliance placed on each of these three legs will depend on the application. Where detection may be delayed for months or years, and recovery may therefore be very difficult—as with bogus bank guarantees—it is prudent to put extra effort into prevention, using techniques such as dual control. Where prevention is difficult to enforce, it is essential that detection be fast enough, and recovery vigorous enough, to provide a deterrent effect. The classic example here is that bank tellers can quite easily take cash, so you need to count the money every day and catch any shortfall by close of business.

Bookkeeping and management control systems are not only one of the earliest security systems, they also have given rise to much of management science and civil law. They are entwined with a company's business processes, and exist in its cultural context. In Swiss banks, two managers' signatures appear on almost everything, while Americans are much more relaxed. In most countries' banks, staff get background checks, can be moved randomly from one task to another, and are required to take holidays at least once a year. But this would be excessive in the typical university department where the opportunities for fraud are much less.

Designing a good bookkeeping system is hard because it's such an interdisciplinary problem. The financial controllers, the personnel department, the lawyers, the auditors, and the systems people all come at the problem from different directions, offer partial solutions, fail to understand each other's control objectives, and things fall down the hole in the middle. Human factors are very often neglected, and systems end up being vulnerable to helpful subordinates or authoritarian managers who can cause dual control to fail. It's important not just to match the controls to the culture, but also to motivate people to use them. For example, in the better-run banks, management controls are marketed to staff as a means of protecting them against blackmail and kidnapping.

Security researchers have so far focused on the small part of the problem, which pertains to creating dual control (or in general, where there are more than two principals, *shared control*) systems. Even this is not at all easy. For example, rule 9 in Clark-Wilson says that security officers can change access rights, so what's to stop a security officer creating logons for two managers and using them to send all the bank's money to Switzerland?

One possible answer is to use cryptography, and split the relevant signing key between two or more principals. In an NT network, the obvious way to manage things is to put users in separately administered domains. With a traditional banking system, using the mainframe operating system MVS, we can separate duties between the system administrator (sysadmin) and the auditor; the former can do anything he wishes, except find out which of his activities the latter is monitoring [95]. But in real life, dual control is hard to do end to end because there are many system interfaces that provide single points of failure; and, in any case, split-responsibility systems administration is tedious.

The practical answer, then, is that most bank sysadmins could do just this type of fraud. Some have tried—where they fall down is that the back-office balancing controls set off the alarm after a day or two, and money laundering controls stop them from getting away with very much. I discuss this further in Section 9.3.2. The point to bear in mind here is that serial controls in the prevent-detect-recover model are usually more important than shared control. They depend ultimately on some persistent state in the system, and are in tension with programmers' desire to keep things simple by making transactions atomic.

There are also tranquility issues. For example, could an accountant, knowing that he was due to be promoted to manager tomorrow, end up doing both authorizations on a large transfer? A technical fix for this might involve a Chinese Wall mechanism supporting a primitive "X may do Y but not Z" ("a manager can confirm a payment only if her name doesn't appear on it as the creator"). In this way, we would end up with a number of exclusion rules involving individuals, groups, and object labels; once the number of rules became large (as it will in a real bank) we would need a systematic way of examining this rule set and verifying that it didn't have any loopholes.

In the medium term, banking security policy—just like medical security policy—may end up finding its most convenient expression using role-based access control; platforms such as Win2K may be heading in this direction. This offers the potential for managing separation of duty policies that involve both parallel elements, such as dual control, and serial elements, such as functional separation along a transaction's path.

A final remark on dual control is that it's often inadequate for transactions involving more than one organization, because of the difficulties of dispute resolution: "My two managers say the money was sent!" "But my two say it wasn't!"

9.2.3 What Goes Wrong

Theft can take a variety of forms, from the purely opportunist to clever insider frauds; but regardless of size most thefts from the average company are due to insiders. There are many surveys. A recent one, by accountants Ernst and Young, reports that 82 percent of the worst frauds in 1999–2000 were committed by employees; nearly half of the perpetrators had been there over five years, and a third of them were managers [697].

Typical computer crime cases include:

- A bank had a system of suspense accounts, which would be used temporarily if one of the parties to a transaction could not be identified (such as when an account number was entered wrongly on a funds transfer). This was a work-

around added to the dual control system to deal with transactions that got lost or otherwise couldn't be balanced immediately. As it was a potential vulnerability, the bank had a rule that suspense accounts would be investigated if they were not cleared within three days. One of the clerks exploited this by setting up a scheme whereby she would post a debit to a suspense account and an equal credit to her boyfriend's account; after three days, she would raise another debit to pay off the first. In almost two years, she netted hundreds of thousands of dollars. (The bank negligently ignored a regulatory requirement that all staff take at least 10 consecutive days' vacation no more than 15 months from the last such vacation.) In the end, she was caught when she could no longer keep track of the growing mountain of bogus transactions.

- A clerk at the Inner London Education Authority wanted to visit relatives in Australia, and to get some money, she created a fictitious school, complete with staff whose salaries were paid into her own bank account. It was discovered only by accident when someone noticed that different records gave the authority different numbers of schools.
- A bank clerk in Hastings, England, noticed that the branch computer system did not audit address changes. He picked a customer who had a lot of money in her account and got a statement only once a year; he then changed her address to his, issued a new ATM card and personal identification number (PIN), and changed her address back to its original value. In total, he stole £8,600 from her account. When she complained, she was not believed: the bank maintained that its computer systems were infallible, and so the withdrawals must have been her fault. The matter was cleared up only when the clerk got an attack of conscience and started stuffing the cash in brown envelopes through the branch's letter box at night. The branch manager finally realized that something was seriously wrong.

All the really large frauds—the cases over a billion dollars—have involved lax internal controls. The collapse of Barings Bank is a good example; there, managers failed to control rogue trader Nick Leeson, as they were blinded by greed for the bonuses his apparent trading profits earned them. The same holds true for other big financial sector frauds, such as the Equity Funding scandal, in which an insurance company's management created thousands of fake people on their computer system, insured them, and sold the policies on to reinsurers; and frauds in other sectors such as Robert Maxwell's looting of the *Daily Mirror* newspaper pension funds in Britain. (For a collection of computer crime case histories, see Parker [602].) Either the victim's top managers were grossly negligent, as in the case of Barings, or were the perpetrators, as with Equity Funding and Maxwell. As a result, a number of standards have been put forward by the accountancy profession, by stock markets, and by banking regulators, about how bookkeeping and internal control systems should be designed. In the United States, for example, there is the *Committee of Sponsoring Organizations* (COSO), a group of U.S. accounting and auditing bodies [196]. I'll return to COSO and explore how to go about designing an internal control system in Chapter 22, "Management Issues," Section 22.4.1.2.

But changing technology also has a habit of eroding controls, which therefore need constant attention and maintenance. For example, thanks to new systems for high-speed processing of bank checks, banks in California will no longer honor requests by

depositors that checks have two signatures. Even when a check has printed on it "Two Signatures Required," banks will honor that check with only one signature [651]. This might seem to be a problem for the customer's security rather than the bank's, but bank checks can also be at risk and if something goes wrong even with a merchant transaction, the bank might still get sued. The vulnerability of shared control to technical attacks continues to grow. Most major accounting packages do not use double-entry bookkeeping internally, but rather create an appearance of it at the presentation layer; and the current trend appears to be toward event databases in which all transactions in an accounting period are accumulated, with reports being generated directly as required. New control strategies may be needed. One possible technical approach is to maintain separate logs of all original events (purchase orders, invoices, payments, etc.) and have programs that constantly cross-check. People-based measures are also highly advisable. Accounts software should empower line managers so that they can monitor their departments' income, expenditure and commitments. Making the technical and managerial controls overlap, so that they cover each others' weaknesses, is the goal; unfortunately, the common outcome is that the technical controls merely duplicate the managerial ones, resulting in common failure modes that fraudsters can exploit.

The lessons to be learned include the following.

- It's not always obvious which transactions are security-sensitive.
- It's hard to maintain a working security system in a changing environment.
- If you rely on customer complaints to alert you to fraud, you had better listen to them.
- There will always be people in positions of relative trust who can get away with a scam—for a while.
- No security policy will ever be completely rigid; there will always have to be workarounds for people to cope with real life, and some of these workarounds will create vulnerabilities.
- It's often hard to tell at first sight whether an exception is due to fraud or to error. So the lower the transaction error rate, the better.

There will always be residual risks. Managing these remains one of the hardest and most neglected of jobs. It requires not just technical measures, such as involving knowledgeable industry experts, auditors, and insurance people in the detailed design, and iterating the design once some loss history is available. It also means training managers, auditors, and others to detect problems and react to them appropriately. I'll revisit this topic in Chapter 22.

The banking industry has gone a long way along this learning curve. The general experience of banks in the English-speaking world is that some 1 percent of staff are fired each year. The typical offense is minor embezzlement, incurring a loss of a few thousand dollars. No one has found an effective way of predicting which staff will go bad; previously loyal staff can be thrown off the rails by shocks such as divorce, or may over time develop a gambling or alcohol habit.

9.3 Wholesale Payment Systems

Systems for transferring money electronically were one of the first applications of the telegraph when it was introduced in the middle of the nineteenth century; and I explained in Chapter 5, "Cryptography," Section 5.2.4 how the system of test keys was developed to compute authentication codes on the messages manually. By the early 1970s, bankers started to realize that a better system was needed:

- The cryptographic vulnerability of the system became apparent.
- Although the test key tables were kept in the safe, it was at least theoretically possible for a bank employee to memorize one of the simpler schemes. With the more complex schemes, even an employee working under close supervision could mentally compute the test on an unauthorized message, while overtly computing the test on an authorized one.
- The schemes didn't support dual control. Although tests were computed by one staff member and checked by another, this doubled the risk rather than halving it. (There are ways to do dual control with manual authenticators, and these had been developed extensively for use in the control of nuclear weapons—I discuss them in Chapter 11, Section 11.4—but this technology was still classified at the time.)
- The major concern was cost and efficiency. There seemed little point in having the bank's computer print out a transaction in the telex room, having a test computed manually, composing a telex to the other bank, checking the test, and then entering it into the other bank's computer. Surely the payments could flow directly from one bank's computer to another?

Clearly, a fresh design was needed.

9.3.1 SWIFT

The Society for Worldwide Interbank Financial Telecommunications (SWIFT) was set up in the 1970s by a consortium of banks to provide a more secure and efficient means of sending payment instructions between member banks. It can be thought of as an email system with built-in encryption, authentication, and nonrepudiation services.

The SWIFT design constraints are interesting. The banks did not wish to trust SWIFT, in the sense of enabling some combination of dishonest employees there to forge transactions. The authenticity mechanisms had to be independent of the confidentiality mechanisms, since at the time a number of countries (such as France) forbade the civilian use of cryptography for confidentiality. The nonrepudiation functions had to be provided without the use of digital signatures, as these hadn't been invented yet. Finally, the banks had to be able to enforce Clark-Wilson type controls over interbank transactions. (Clark-Wilson also hadn't been invented yet, but its components—dual control, balancing, audit, and so on—were well enough established.)

The SWIFT design is summarized in Figure 9.2. Authenticity of messages was assured by computing a message authentication code (MAC) at the sending bank and checking it at the receiving bank. Formerly, the keys for this MAC were managed end-to-end: whenever a bank set up a relationship overseas, the senior manager who negotiated it would exchange keys with her opposite number, whether in a face-to-face

meeting or afterward by post to each other's private addresses. There would typically be two key components to minimize the risk of compromise, with one sent in each direction (on the grounds that even if a bank manager's mail is stolen from her mailbox by a criminal at one end, it's not likely to happen at the other end as well). The key would not be enabled until both banks confirmed that it had been safely received and installed.

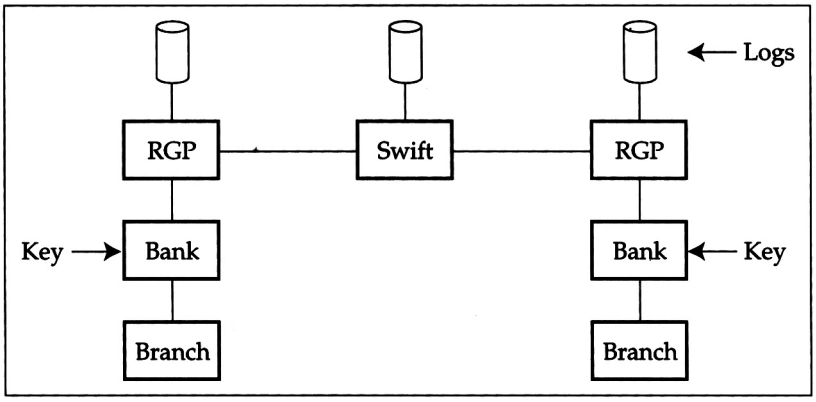

Figure 9.2 Architecture of SWIFT.

This way, SWIFT had no part in the message authentication. As long as the authentication algorithm SWIFT chose was sound, none of their staff could forge a transaction. (The authentication algorithm used is supposed to be a trade secret; but because banks like their security mechanisms to be international standards, a natural place to look might be the algorithm described in ISO 8731 [657].) In this way, they got the worst of all possible worlds: the algorithm was fielded without the benefit of public analysis but got it later once it was expensive to change. (An attack was found on the ISO 8731 message authentication algorithm and published in [621], but the number of messages required to break it is too large for a practical attack on a typical system that is used prudently.)

Although SWIFT itself was largely outside the trust perimeter for message authentication, it did provide a nonrepudiation service. Banks in each country sent their messages to a *regional general processor* (RGP), which logged them and forwarded them to SWIFT, which also logged them and sent them on to the recipient bank via the RGP in its country, which also logged them. The RGPs were generally run by different facilities management firms. Thus, a bank (or a crooked bank employee) wishing to dishonestly repudiate a done transaction—or claim that one had been done when it hadn't—would have to subvert not just SWIFT itself, but also two independent local contractors (in order to alter their log entries). Logs can be a powerful evidential resource, and are much easier for judges to understand than cryptography.

Confidentiality depended on line encryption devices between the banks and the RGP node, and between these nodes and the main SWIFT processing sites. Key management was straightforward. Keys were hand-carried in EEPROM cartridges between the devices at either end of a leased line. In countries where confidentiality was illegal, these devices could be omitted without impairing the authenticity and nonrepudiation mechanisms.

Dual control was provided either by the use of specialized terminals (in small banks) or by mainframe software packages that could be integrated with a bank's main production system. The usual method of operation is to have three separate staff to do a SWIFT transaction: one to enter it, one to check it, and one to authorize it. (As the checker can modify any aspect of the message, this really gives only dual control, not triple control; and the programmers who maintain the interface can always attack the system there). Reconciliation was provided by checking transactions against daily statements received electronically from correspondent banks. This meant that someone who managed to get a bogus message into the system would sound an alarm within two or three days.

9.3.2 What Goes Wrong

SWIFT I ran for 20 years without a single report of external fraud. In the mid-1990s, it was enhanced by the addition of public key mechanisms. MAC keys are now shared between correspondent banks using public key cryptography, and the MACs themselves may be further protected by a digital signature. The key management mechanisms have been ensconced as ISO standard 11166, which in turn has been used in other systems (such as CREST, which is used by banks and stockbrokers to register and transfer U.K. stocks and shares). There has been some debate over the security of this architecture [47, 657]: Quite apart from the centralization of trust brought about by the adoption of public key cryptography—in that the central certification authority can falsely certify a key as belonging to a bank when it doesn't—CREST (at least) adopted public keys that are too short (512 bits). At least one RSA public key of this length has been factored surreptitiously by a group of students.

However, the main practical attacks on such systems have not involved the payment system mechanisms themselves. The typical attack comes from a bank programmer inserting a bogus message into the processing queue. It usually fails because he does not understand the other controls in the system or the procedural controls surrounding large transfers. For example, banks typically keep mutual overdraft limits of perhaps a million dollars, so transfers of larger amounts need the prior involvement of the foreign exchange dealers; there's the daily back-office reconciliation; money-laundering laws require staff to report large cash withdrawals; and anyone who opens a bank account, receives a large incoming wire transfer, then starts frantically moving money out again will need a very convincing excuse. Consequently, the programmer who inserts a bogus transaction into the system usually gets arrested when he turns up to collect the cash.

Other possible technical attacks—such as inserting Trojan software into the PCs used by bank managers to initiate transactions, wiretapping the link from the branch to the bank mainframe, subverting the authentication protocol used by bank managers to log on, and even inserting a bogus transaction in the branch LAN to appear on the relevant printer—would also run up against these controls.

In fact, most large-scale bank frauds that "worked" have not used technical attacks but exploited procedural vulnerabilities, such as the following:

- The classic example is a letter of guarantee. It is common enough for a company in one country to guarantee a loan to a company in another. This can be set up as a SWIFT message or even a paper letter. But as no cash changes

hands at the time, the balancing controls are inoperative. If a forged guarantee is accepted as genuine, the "beneficiary" can take his time borrowing money from the accepting bank, laundering it, and disappearing. Only when the victim bank realizes that the loan has gone sour, and tries to call in the guarantee, is the forgery discovered.

- An interesting fraud of a slightly different type took place in 1986 between London and Johannesburg. At that time, the South African government operated two exchange rates, and in one bank the manager responsible for deciding which rate applied to each transaction conspired with a rich man in London. They sent money out to Johannesburg at an exchange rate of seven Rand to the Pound, and back again the following day at four. After two weeks of this, the authorities became suspicious, and the police came round. On seeing them in the dealing room, the manager fled without stopping to collect his jacket, drove over the border to Swaziland, and flew via Nairobi to London. There, he boasted to the press about how he had defrauded the wicked apartheid system. As Britain has no exchange control, exchange control fraud isn't an offense, so he couldn't be extradited. The conspirators got away with millions, and the bank couldn't even sue them.
- Perhaps the best-known funds transfer fraud occurred in 1979 when Stanley Rifkin, a computer consultant, embezzled over $10 million from Security Pacific National Bank. He circumvented the money-laundering controls by agreeing to buy a large shipment of diamonds from a Russian government agency in Switzerland. He got the transfer into the system by observing an authorization code used internally when dictating transfers to the wire transfer department, and simply used it over the telephone (a classic example of dual control breakdown at a system interface). He even gave himself extra time to escape by doing the deal just before a U.S. bank holiday. Where he went wrong was in not planning what to do after he collected the stones. If he had hidden them in Europe, gone back to the United States, and helped investigate the fraud, he might well have got away with it; as it was, he ended up on the run and got caught.

The moral is that we must always be alert to things which defeat separation-of-duty controls by introducing a single point of failure. Even if we can solve the technical problems of systems administration, interfaces, and so on, there's still the business system analysis problem of what we control—quite often, critical transactions aren't obvious to a casual inspection.

9.4 Automatic Teller Machines

Another reason that dual control—although necessary—is not sufficient, emerges from the study of "phantom withdrawals"—complaints of unauthorized cash withdrawals from *automatic teller machines* (ATMs).

ATMs, also known as cash machines, have been one of the most influential technological innovations of the twentieth century. Quite apart from their social and economic impact, they are just as important to the security engineer both as a source of technology and as a case study.

ATMs were the first large-scale retail transaction processing systems. They have been around since 1968; the world installed base is now about 500,000 machines. The technology developed for them is now also used in terminals for *electronic funds transfer at the point of sale* (EFTPOS, or just POS) in shops. Modern block ciphers were first used on a large scale in ATM networks, to generate and verify PINs in secure hardware devices located within the ATMs and at bank computer centers. This technology, including block ciphers, tamper-resistant hardware, and the supporting protocols, ended up being used in many other applications, from postal franking machines to lottery ticket terminals. ATMs were the "killer app" that got modern commercial cryptology off the ground.

9.4.1 ATM Basics

Many ATMs operate using some variant of a system developed by IBM for its 3614 series cash machines in the mid-1970s. This uses a secret key, called the *PIN key*, to encrypt the account number, then decimalize it and truncate it. The result of this operation is called the *natural PIN*; an offset can be added to it to give the PIN the customer must enter. The offset has no real cryptographic function; it just enables customers to choose their own PIN. An example of the process is shown in Figure 9.3.

Dual control is implemented in this system using tamper-resistant hardware. A cryptographic processor, often called a *security module*, is kept in the bank's central computer room. It will perform a number of defined operations on customer PINs and on related keys in such a way that:

- Operations on the clear values of customer PINs, and on the keys or other material needed to compute them or used to protect them, are all done in tamper-resistant hardware and the clear values are never made available to any single member of the bank's staff.
- Thus, for example, the cards and PINs are sent to the customer via separate channels. The cards are personalized in a facility with embossing and mag-strip printing machinery; the PIN mailers are printed in a separate facility containing a printer attached to a security module.
- A *terminal master key* is supplied to each ATM in the form of two printed components, which are carried to the branch by two separate officials, input at the ATM keyboard, and combined to form the key. Similar procedures are used to set up keys between banks and network switches such as VISA.
- If ATMs are to perform PIN verification, the PIN key is encrypted under the terminal master key, then sent to the ATM.
- If the PIN verification is to be done centrally over the network, the PIN is encrypted under a key that is set up using the terminal master key, and sent from the ATM to the security module for checking.
- If the bank's ATMs are to be networked with other banks', then one uses transactions that will take an encrypted PIN from one source (such as encrypted under an ATM key), decrypt it, and re-encrypt it for its destination (such as using a key shared with VISA). This *PIN translation* function is done entirely within the hardware security module, so that clear values of PINs are never available to the bank's programmers.

Account number N (on the mag stripe):	8807012345691715
PIN key KP:	FEFEFEFEFEFEFEFE
Result of DES $\{N\}_{KP}$:	A2CE126C69AEC82D
$\{N\}_{KP}$ decimalized:	0224126269042823
Natural PIN:	0224
Offset:	6565
Customer PIN:	6789

Figure 9.3 IBM method for generating bank card PINs.

During the 1980s and 1990s, the hardware security modules became more and more complex, as ever more functionality got added. An example of a leading product in 2000 is the IBM 4758, this also has the virtue of having its documentation available publicly online for study (see [397] for the command set, and [718] for the architecture and hardware design). I'll discuss this in Chapter 14, "Physical Tamper Resistance."

But extending the dual control security policy from a single bank to tens of thousands of banks worldwide, as modern ATM networks do, proved not to be completely straightforward:

- When people started building ATM networks in the mid-1980s, many banks used software encryption rather than hardware security modules to support the machines. So in theory, any bank's programmers might get access to the PINs of any other bank's customers. The remedy was to push through standards for security module use. In many countries (such as the United States), these standards were largely ignored; but even where they were respected, some banks continued using software for transactions involving their own customers. So some keys (such as those used to communicate with ATMs) had to be available in software, too, and knowledge of these keys could be used to compromise the PINs of other banks' customers. Consequently, the protection given by the hardware TCB was rarely complete.
- It is not feasible for 10,000 banks to share keys in pairs, so each bank connects to a switch provided by an organization such as VISA or Cirrus, and the security modules in these switches translate the traffic. The switches also do accounting, and enable banks to settle their accounts for each day's transactions with all the other banks in the system, by means of a single electronic debit or credit. The switch is highly trusted; if something goes wrong, the consequences could be severe. In one case, there turned out to be not just security problems but also dishonest staff. The switch manager ended up a fugitive from justice, and the bill for remediation was in the millions.
- Corners are cut to reduce the cost of dealing with huge transaction volumes. For example, it is common for authentication of authorization responses to be turned off. The effect is that anyone with access to the network can cause a given ATM to accept any card presented to it, simply by replaying a positive authorization response. Network managers claim that should a fraud ever start, the authentication can always be turned back on. This might seem reasonable; attacks involving manipulated authorization responses are very rare. But such shortcuts—even when reasonable on grounds of risk and cost—mean that a

bank that claims, in response to a customer dispute, that its ATM network cannot possibly be attacked, and so the transaction must be the customer's fault, is not telling the truth. What's more, turning on the message authentication codes suddenly in response to a fraud could be difficult. Some banks' implementations might not support them properly or at all, and performance degradation might result unless more encryption devices are installed rapidly. One is reminded of the saying that 'optimization is the process of taking something which works, and replacing it by something which doesn't quite but is cheaper'.

There are many other ways in which ATM networks can be attacked in theory. For example, they mostly use single-key DES encryption, even for top-level keys, and DES can now be broken by exhaustive keysearch. However, one of the interesting things about these systems is that they have now been around long enough, and have been attacked enough by both insiders and outsiders, to give us a lot of data points on how such systems fail in practice.

9.4.2 What Goes Wrong

ATM fraud is an interesting study, as the ATM system is mature, with huge volumes and a wide diversity of operators. An extensive survey can be found in [19], and further material in [20]. Here, I'll summarize the more important and interesting points.

The engineers who designed ATM security systems in the 1970s and 1980s (of whom I was one) assumed that criminals would be relatively sophisticated, fairly well informed about the system design, and rational in their choice of attack methods. In addition to worrying about the many banks that were slow to buy security modules, and about the implementation loopholes such as omitting authentication codes on authorization responses, we agonized over whether the encryption algorithms were strong enough, and whether the tamper-resistant boxes were resistant enough. We were afraid that a maintenance engineer could disable the tamper sensing circuitry on one visit, and extract the keys on the next. We worried whether the random-number generators used to manufacture keys were random enough. And a very serious concern was that we just couldn't enforce dual control properly. Bank managers considered it beneath their dignity to touch a keyboard, so rather than entering the ATM master key components themselves after a maintenance visit, most of them would just give both key components to the ATM engineer. We believed that sooner or later a repairman would get his hands on a bank's PIN key, forge cards in industrial quantities, close down the whole system, and wreck public confidence in electronic banking.

The bulk of the actual phantom withdrawals, however, have one of the following three simple causes:

- *Simple processing errors account for a lot of disputes.* With U.S. customers making something like 5 billion ATM withdrawals a year, even a system that makes only one error per 100,000 transactions will give rise to 50,000 disputes a year. In practice, the error rate seems to lie somewhere between 1 in 10,000 and 1 in 100,000. One source of errors we tracked down was that a large bank's ATMs would send a transaction again if the network went down before a confirmation message was received from the mainframe; periodically, the

mainframe itself crashed, and "forgot" about open transactions. We also found customers whose accounts were debited with other customers' transactions, and other customers who were never debited at all for their card transactions. (We used to call these cards "directors' cards," and joked that they were issued to bank directors.)

- *Thefts from the mail are also huge.* They are reckoned to account for 30 percent of all U.K. payment card losses, but most banks' postal control procedures are dismal. For example, in February 1992, I asked my bank for an increased card limit: the bank sent not one, but two, cards and PINs through the post. These cards arrived only a few days after intruders had got hold of our apartment block's mail and torn it up looking for valuables. It turned out that this bank did not have the systems to deliver a card by registered post. (I'd asked them to send the card to the branch for me to pick up, but someone at the branch had simply readdressed the envelope to me.) Since then, many banks have found that better postal controls are the one way they can make enough of a dent in their fraud rates to affect their bottom line.
- *Frauds by bank staff appear to be the third major cause of phantoms.* I mentioned the Hastings case in Section 9.2.3; there are many others. For example, in Paisley, Scotland, an ATM repairman installed a portable computer inside an ATM to record customer card and PIN data, then went on a spending spree with forged cards. In London, England, a bank stupidly used the same cryptographic keys in its live and test systems; maintenance staff found out that they could work out customer PINs using their test equipment, and started offering this as a service to local criminals at £50 a card. Such frauds are particularly common in countries such as Britain, where banks had for many years a policy of denying that their cash machines could possibly make an error. Bank staff knew that customer complaints would be stonewalled rather than investigated.

These failures are all very much simpler and more straightforward than the ones we engineers had worried about. In fact, the only fraud we had anticipated, and that happened to any great extent, came from the practice (common in the 1980s) of letting ATMs process transactions while the network was down or the central mainframe was offline. Though this was convenient—it meant 24-hour service—criminals, especially in Italy and England, learned to open bank accounts, duplicate the cards, then use them to withdraw money simultaneously from a large number of ATMs overnight when the network was down [494]. Such frauds led most banks to make ATM operation online-only by the mid-1990s.

However, there were numerous frauds that happened in quite unexpected ways. We already mentioned the Utrecht case in Section 2.8, where a tap on a garage point-of-sale terminal was used to harvest card and PIN data; and the "encryption replacement" trick by which banks that just encrypted the customer PIN and wrote it on the customer card enabled crooks to change the account number on their own card to somebody else's. There were many more.

- A favorite modus operandi was for villains to stand in ATM queues, observe customers' PINs, pick up the discarded ATM tickets, copy the account numbers from the tickets to blank cards, and use these to loot the customers' accounts. This trick was first reported in New York in the mid-1980s; it was still working in the San Francisco Bay Area in the mid-1990s. Yet there are many

simple countermeasures, such as incorporating extra data on the mag strip, or just not printing the full account number on the ticket.

- One bank's systems had this feature: when a telephone card was entered at an ATM, it believed that the previous card had been inserted again. Crooks stood in line, observed customers' PINs, and helped themselves. This seems to have been an obscure programming error involving the card reader's error handler; one can't expect all such errors to be found during testing.
- One make of ATM would output 10 banknotes from the lowest-denomination nonempty cash drawer whenever a certain 14-digit sequence was entered at the keyboard. One bank printed this sequence in its branch manual, and three years later there was a sudden spate of losses. These went on until all the banks using the machine put in a software patch to disable the transaction.
- One small institution issued the same PIN to all its customers, as a result of a simple programming error.
- Several banks thought up check-digit schemes to enable PINs to be checked by offline ATMs and point-of-sale devices without these devices having a full encryption capability. For example, customers of one British bank would get a credit card PIN with digit 1 plus digit 4 equal to digit 2 plus digit 3, and a debit card PIN with 1 plus 3 equals 2 plus 4. This meant that crooks could use stolen cards in offline devices by entering a PIN such as 4455.
- Some banks show a complete disregard for prudent procedure. In August 1993, my wife went into a branch of our bank with a witness and said that she'd forgotten her PIN. The teller helpfully printed her a new PIN mailer from a printer attached to a PC behind the counter. There were no visible dual controls. Worse, this was not the branch where our account is kept. Nobody knew her and the only identification she offered was our bank card and her checkbook. When procedural controls are so lax that anyone can walk in off the street and get a PIN for a random customer account, no amount of encryption technology will do much good. (The bank in question has since fallen victim to a takeover.)
- A rapidly growing modus operandi is to use false terminals to collect customer card and PIN data. Attacks of this kind were first reported from the United States in 1988; there, crooks built a vending machine that would accept any card and PIN, and dispense a packet of cigarettes. They put their invention in a shopping mall, and harvested PINs and magnetic strip data by modem. In 1993, two villains installed a bogus ATM in the Buckland Hills Mall in Connecticut [421, 590]. They had managed to get a proper ATM and a software development kit for it—all bought on credit. Unfortunately for them, they decided to use the forged cards in New York, where cash machines have hidden video cameras; they ended up getting long stretches in Club Fed. The largest and most recent case to date took place in 1999 in Canada. This involved doctored point-of-sale terminals, and led to the arrest of dozens of alleged Eastern European organized-crime figures in the Toronto area and elsewhere [54, 91].

In conclusion, the main thing we did wrong when designing ATM security systems in the early to mid-1980s was to worry about criminals being clever; we should rather have worried about our customers—the banks' system designers, implementers, and testers—being stupid.

Crypto is usually only part of a very much larger system. It gets a lot of attention because it is mathematically interesting; but as correspondingly little attention is paid to the "boring" bits such as training, usability, standards, and audit, it's rare that the bad guys have to break the crypto to compromise a system. It's also worth bearing in mind that there are so many users for large systems, such as ATM networks, that we must expect the chance discovery and exploitation of accidental vulnerabilities that were simply too obscure to be caught in testing.

9.4.3 Practical Implications

In some countries (including the United States), the banks have to carry the risks associated with new technology. Following a legal precedent, in which a bank customer's word that she had not made a withdrawal was found to outweigh the banks' experts' word that she must have done so [427], the U.S. Federal Reserve passed Regulation E, which requires banks to refund all disputed transactions unless they can prove fraud by the customer [276]. This has led to some minor abuse—misrepresentations by customers are estimated to cost the average U.S. bank about $15,000 a year—but this is an acceptable cost (especially as losses from vandalism are typically three times as much) [813].

In other countries—such as Britain and Norway—the banks got away for many years with claiming that their ATM systems were infallible. Phantom withdrawals, they maintained, could not possibly occur, and a customer who complained of one must be mistaken or lying. This position was finally demolished (in the Britain at least) when significant numbers of criminals were jailed for ATM fraud, and the problem couldn't plausibly be denied any more. (A number of these cases are described in [19, 20].) Until that happened, however, there were some rather unpleasant incidents that got banks a lot of bad publicity. Perhaps the worst was the Munden case.

John Munden was one of our local police constables, based in Bottisham, Cambridgeshire; his beat included the village of Lode where I lived at the time. He came home from holiday in September 1992 to find his bank account empty. He asked for a statement, found six unexpected withdrawals for a total of £460 (then about $700), and complained. His bank responded by having him prosecuted for attempting to obtain money by deception. It came out during the trial that the bank's system had been implemented and managed in a ramshackle way; the disputed transactions had not been properly investigated; and all sorts of wild claims were made by the bank, such as that its ATM system couldn't suffer from bugs as its software was written in Assembler. Nonetheless, it was basically the constable's word against the bank's. He was convicted in February 1994 and fired from the police force.

This miscarriage of justice was overturned on appeal, and in an interesting way. Just before the appeal was due to be heard, the prosecution served up a fat report from the bank's auditors claiming that the system was secure. The defense demanded equal access to the bank's systems for its own expert. The bank refused, and the court therefore disallowed all the bank's computer evidence—including its bank statements. The ap-

peal succeeded, and Munden got reinstated. But this was only in July 1996—he'd spent the better part of four years in limbo, and his family had suffered terrible stress. Had the incident happened in California, he could have won enormous punitive damages, a point bankers should ponder as their systems become global and their customers can be anywhere.

The lesson to be drawn from such cases is that dual control is not enough. If a system is to provide evidence, it must be able to withstand examination by hostile experts. In effect, the bank in the Munden case had used the wrong security policy. What it really needed wasn't dual control, but *nonrepudiation:* the ability for the principals in a transaction to prove afterward what happened. This could have been provided by installing ATM cameras; although these were available (and are used in some U.S. states), they were not being used in Britain.

The issue of nonrepudiation arises in a number of other applications. Often, the right question to ask is not about the mechanism (cameras, biometrics, digital signatures, . . .) but about the motive. Why should a U.K. bank have spent money on ATM cameras that would have undermined its infallibility policy? (One English bank did install ATM cameras during the spate of phantom withdrawals, but took them out again under pressure from the other banks.) And why for that matter should people shopping on the Net use digital signatures, if these will just make it harder to deny a transaction when things go wrong? We will revisit this issue again and again in later chapters.

9.5 Summary

Banking systems are interesting in a number of ways.

Bookkeeping applications give us a mature example of systems whose security is oriented toward authenticity and accountability rather than confidentiality. Their protection goal is to prevent and detect frauds being committed by dishonest insiders. The Clark-Wilson security policy provides a model of how they operate. It can be summarized as:

> *All transactions must preserve an invariant of the system, namely that the books must balance (so a negative entry in one ledger must be balanced by a positive entry in another one); some transactions must be performed by two or more staff members; and records of transactions must not be destroyed after they are committed.*

This was based on time-honored bookkeeping procedures, and led the research community to consider systems other than variants of Bell-LaPadula.

But manual bookkeeping systems use more than just dual control. Although some systems do need transactions to be authorized in parallel by two or more staff, a separation of duty policy more often works in series, in that different people do different things to each transaction as it passes through the system. Designing bookkeeping systems that do this effectively is a major problem which is often neglected and which involves input from many disciplines. Another common requirement is nonrepudiation—that principals should be able to generate, retain, and use evidence about the relevant actions of other principals.

The other major banking application, remote payment, is increasingly critical to e-commerce. In fact, wire transfers of money go back to the middle of the Victorian era. Because there is an obvious motive to attack these systems, and villains who steal large amounts and get caught are generally prosecuted, payment systems are a valuable source of information about what goes wrong. Their loss history teaches us the importance of minimizing the background error rate, preventing procedural attacks that defeat technical controls (such as thefts of ATM cards from the mail), and having adequate controls to deter and detect internal fraud.

Payment systems have also played a significant role in the development and application of cryptology. One innovation was the idea that cryptography could be used to confine a critical part of the application to a trusted computing base consisting of tamper-resistant processors—an approach since used in many other applications.

Research Problems

Designing transaction sets for bookkeeping applications is still pre-scientific; we could do with tools to help us do it in a more systematic, less error-prone way. Accountants, lawyers, financial market regulators, and system engineers all seem to feel that this is someone else's responsibility. This is a striking opportunity to do multidisciplinary research that might actually be useful.

At an even more basic level, we don't even fully understand stateful access control systems, such as Clark-Wilson and Chinese Wall. To what extent does one do more than the other on the separation-of-duty front? How should dual control systems be designed anyway? How much of the authorization logic can we abstract out of application code into middleware? Can we separate policy and implementation to make enterprise-wide policies easier to administer?

There are some useful distinctions, such as policy versus mechanism versus management, push versus pull, and specification versus runtime controls. There are some prototype engines for enforcing an arbitrary policy—such as HP's authorization server product [772] and AT&T's Policymaker [115]. Developing such engines to deal with the full generality of possible security policies is still an open problem.

As for robustness of cryptographic systems, the usability of security mechanisms, and assurance generally, these are huge topics that are still only partially mapped. Robustness and assurance are partially understood, but usability is still a very gray area. There are many more mathematicians active in security research than applied psychologists, and it shows.

Further Reading

I don't know of a comprehensive book on banking computer systems, although there are many papers on specific payment systems available from the Bank for International Settlements [72]. When it comes to developing robust management controls and business processes that limit the amount of damage that any one staff member can do, there is a striking lack of hard material (especially given the need that new e-businesses have for such systems). There was one academic conference in 1997 [416]; but the business

books that touch on these issues all seem to focus on financial management and on the soft aspects of management control such as "tone at the top." I'll revisit this in Chapter 22.

For the specifics of financial transaction processing systems, the cited articles [19, 20] provide a basic introduction. More comprehensive, if somewhat dated, is [221], while [336] describes the CIRRUS network as of the mid-'80s. The most informative public domain source—though somewhat heavy going—is probably the huge online manuals for the equipment in question, such as the IBM 4758 and CCA [397].