Ross J. Anderson

Why Cryptosystems **Fail**

Cryptography is used by governments, banks, and other organizations to keep messages secret and to protect electronic transactions from modification. It is basically an engineering discipline, but differs in a rather striking way from, for example, aeronautical engineering: there is almost no public feedback about how cryptographic systems fail.

Commercial airline crashes are extremely public events. Investigators rush to the scene, and their inquiries involve experts from a wide range of interests—from the carrier, through the manufacturer, to the pilots' union. Their findings are examined by journalists and politicians, discussed on electronic bulletin boards and in pilots' messes, and passed on by flying instructors. This learning mechanism is the main reason why, despite the inherent hazards of flying, the risk of an individual being killed on a scheduled air journey is only about one in a million.

Cryptographers rarely get this kind of feedback, and indeed the history of the subject shows the same mistakes being made over and over again. For example, Norway's rapid fall in the Second World War was largely due to the Germans' success in solving British naval codes—using exactly the same techniques that the Royal Navy's own "Room 40" had used against Germany in the previous war [16].

Although we now have a reasonable history of cryptology up to the end of World War II, a curtain of silence has descended on government-sector use of this technology since then. Although this is not particularly surprising, it does leave a large experience deficit, especially as the introduction of computers since 1945 has changed the situation considerably. It is as if accident reports were only published for piston-engined aircraft, while the causes of all jet-aircraft crashes were kept a state secret.

This secrecy is no longer appropriate, as military messaging networks now make up only about 1% of the world's cryptography, whether we measure this by the number of users or by the number of terminals. There are some civilian applications for secure messaging, such as interbank money transfer and burglar alarm signaling; but the great majority of fielded cryptographic systems are in applications such as bank cards, pay-TV, road tolls, office building and computer access tokens, lottery terminals, and prepayment electricity meters. Their job is basically to make petty crime, such as card forgery, slightly more difficult.

Cryptography was introduced to the commercial world from the military by designers of automatic teller machine (ATM) systems in the early 1970s. Since then, ATM security techniques have inspired many of the other systems—especially those where customers initiate low-value transactions for which we want to account. One might therefore expect the ATM experience to give a good first-order threat model for cryptographic systems in general.

Automatic Teller Machine Disputes

In some countries, banks are responsible for the risks associated with new technology. In 1980, a New York court believed a bank customer's word that

©YCO LING STUDIO

she had not made a withdrawal, rather than the word of the bank's experts that she must have done so [15]; the Federal Reserve then passed regulations that require U.S. banks to refund all disputed electronic transactions unless they can prove fraud by the customer. Since then, many U.S. ATM cash dispensers have had video cameras installed.

In Britain, the courts have not yet been so demanding; despite a parliamentary commission that found that the personal identification number (PIN) system was insecure [14], bankers simply deny that their systems can ever be at fault. Customers who complain about "phantom withdrawals" are told that they must be lying, or mistaken, or that they must have been defrauded by their friends or relatives. This has led to a string of court cases in the U.K.:

- A teenage girl in Ashton was convicted in 1985 of stealing £40 from her father. She pleaded guilty on the advice of her lawyers that she had no defense, and then disappeared; it later turned out that there had never been a theft, but a clerical error by the bank, which tried to cover it up.
- A Sheffield police sergeant was charged with theft in November 1988 after a phantom withdrawal took place on a card he had confiscated from a suspect. He was lucky: his colleagues located the person who made the transaction after the disputed one, and her testimony cleared him.
- Charges of theft against an elderly woman in Plymouth were dropped after our inquiries showed the bank's computer security systems were a shambles. The same happened in a case against a taxi driver in Great Yarmouth.
- After a police constable complained he had not made six ATM withdrawals that appeared on his bank statement, the bank had him prosecuted and convicted for attempting to obtain money by deception. Their technical evidence was highly suspect; there was an outcry in the press, and an appeal is under way.
- Customers are suing banks in the civil courts in both England and Scotland, and a case may be launched shortly in Norway as well.

We have been involved in providing expert advice in many of these cases, which produced a vast quantity of evidence. In addition to this, and other information discovered through the legal process, we have interviewed former bank employees and criminals, searching the banking, legal, and technical literatures, and drawn on experience gained designing cryptographic equipment. One outcome of all this activity has been the first unclassified study of how and why cryptosystems fail.

The Three Common Problems with ATM Security

Automatic teller machine systems use encryption to protect customers' PINs. The details vary from one bank to another, but many use variants of a system originally developed by IBM [21], in which the PIN is derived from the account number by encryption. It is also encrypted while being sent from the ATM to the bank for verification.

When the crypto know-how was originally imported from the defense sector, a threat model came with it. This model presumed that attacks on the system would be technically sophisticated, and might involve cryptanalysis or the manipulation of transactions at some point in the path between the ATM and the bank that issued the card.

Indeed, there are many ways in which a skilled attacker could penetrate the world's ATM networks [3]. Some networks do not encrypt PINs properly, or at all; many banks, especially in the U.S., do encryption in software rather than hardware, with the result that the keys are known to programmers; some older ATMs and encryption devices have known weaknesses; and even those systems that use approved encryption hardware can be vulnerable, as the Data Encryption Standard algorithm is becoming increasingly open to attack [24]. All these facts are used by encryption equipment sales staff in their efforts to persuade bankers to buy the latest products.

Of the hundreds of documented failures of ATM security, however, only two involved such attacks: in one, a telephone engineer in Japan recorded customer card data and PINs from a phone line; in the other, technicians programmed a communications processor to send only positive authorizations to an ATM where accomplices were waiting. None of the other thefts and frauds were due to skilled attack, but were rather made possible by errors in the design or operation of the ATM system itself.

The three main causes of phantom withdrawals did not involve cryptology at all: they were program bugs, postal interception of cards, and thefts by bank staff.

First, there is a "background noise" of transactions that turn out to be wrongly processed (e.g., posted to the wrong account). It is well known that it is difficult to get an error rate below about 1 in 10,000 on large, heterogeneous transaction processing systems such as ATM networks [10]; yet, before the British litigation started, the government minister responsible for Britain's banking industry was claiming an error rate of 1 in 1.5 million! Under pressure from lawyers, this claim was trimmed to 1 in 250,000, then 1 in 100,000, and most recently to 1 in 34,000. Even this last figure would still mean that about 30,000 phantom withdrawals a year in Britain (and over 200,000 in the United States) are caused by processing errors.

Second, problems with the postal service are also well known and can be particularly acute in university towns. In Cambridge, for example, approximately 4,000 people open bank accounts every October; their ATM cards and PINs are delivered to college pigeonholes, which are wide open to thieves. Yet this author's bank was unable to arrange for a card to be sent by recorded delivery; its system designers had not anticipated the requirement.

The third big problem is theft by bank staff. British banks dismiss about 1% of their staff every year for disciplinary reasons, and many of these firings are for petty thefts in which ATMs can easily be involved. There is a moral hazard here: staff know that many ATM-related thefts go undetected because of the policy

of denying that they are even possible. We discovered a number of cases:

- A woman from Hastings had £8,600 (about $13,000) stolen from her bank account by a bank clerk who issued an extra card for it. The bank's systems did not control address changes, so the clerk was able to change her address to his, issue a new card, and change it back again. When she finally noticed the loss and complained, she was not believed; the thief was only discovered because he suffered an attack of conscience and confessed.
- Technical staff also steal clients' money, knowing that complaints will probably be ignored. At one bank in Scotland, a maintenance engineer fitted an ATM with a hand-held computer, which recorded customers' PINs and account numbers. He then produced counterfeit cards and looted the accounts.
- At another bank, the live and test systems were set up with the same set of keys, which meant that the operators could work out PINs on their test equipment. Eventually, some of them started charging the local underworld £50 a time to calculate PINs for stolen cards; and when the bank's security manager discovered what was going on, he was killed in an accident in which organized crime may have been involved. The bank did not bother to send its customers new cards.
- One type of ATM had a test transaction that output 10 bank notes when a 14-digit sequence was entered at the keyboard, and a bank that used these machines printed this sequence in its branch operations manual. Three years later, there was a sudden series of losses, which went on until all the banks using the machine put in a software patch to disable this feature.
- In another bank, a protégé of the deputy managing director sent a circular to all branches announcing that to cut costs, a number of dual-control procedures were being abolished, including that on handling ATM cards and PINs. Losses increased tenfold; but managers in the affected departments were unwilling to risk their careers by making a fuss.

Most thefts by staff appear as phantom withdrawals at ATMs in the victim's neighborhood. British banks maintain that a computer security problem would result in a random distribution of transactions around the country, and as most disputed withdrawals happen near the customer's home or place of work, these must be due to card-holder negligence. Customer complaints should be a bank security manager's prime source of information; but in Britain they often end up reinforcing corporate complacency.

The problem is not limited to Britain. While some countries (such as Denmark and Singapore) follow the U.S. example and give the benefit of the doubt to the customer, there are more (such as Norway, Portugal, and Italy) that follow the British model, and others (such as Germany and Holland) that sit uncomfortably in the middle.

More Exotic Attacks

We discovered a number of external attacks, which exploited a wide range of design and implementation blunders.

- Two men were convicted at Winchester Crown Court of a simple but effective scam. They would stand in lines of ATM customers, 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 by no means new. It was first reported almost 10 years ago when a former employee of a New York bank stole over $80,000; and most recently, in May 1994, two men were arrested for spying on ATMs in San Jose with a video camera. The attack works when the full account number is printed on the ATM ticket, and is clearly trivial to prevent; but in Britain, it took persistent press publicity to get the banks to fix the problem [18].

- One British bank's cash machines were fooled by telephone cards. When one of these was inserted, the ATM believed that the previous card had been put in again. Thieves stood in line, observed customers' PINs, and helped themselves to cash.
- In early 1992, another British bank sent each of its card holders a letter warning them of the dangers of writing down their PIN, and suggesting instead that they conceal the PIN by encoding it on a distinctive piece of squared cardboard, which was designed to be kept alongside the ATM card. Suppose your PIN is 2256. Choose a four-letter word, say "blue." Write these four letters down in the second, second, fifth, and sixth columns of the grid, respectively, as shown in Figure 1.

1	2	3	4	5	6	7	8	9	0
	b								
	l								
				u					
					e				

Figure 1. How not to encrypt PINs

Now fill up the empty boxes with random letters. Easy, isn't it? Of course, there might be only two dozen four-letter words that can be made up from a given grid of random letters, so a thief's chance of being able to use a stolen card within three tries has just gone from 1 in 3,333 to 1 in 8.

- One small upper-crust private bank belied its exclusive image by giving all its customers the same PIN. This was a simple programming error; but in another, more down-market institution, a programmer deliberately arranged things so that only three different PINs were issued, with the idea that this would provide his personal pension fund. In neither case was the problem detected until quite some time had passed: the PIN mailers were subjected to strict handling precautions, so no one in the bank had the opportunity to notice

the problem (and it did not occur to anyone to check).

- One of the largest London banks had written the encrypted PIN on the card's magnetic strip. The criminal fraternity found by trial and error that you could change the account number on your own card's magnetic strip to that of your target, and then use it with your own PIN to loot the targeted account. A document about this technique circulated in the British prison system, and two men were recently charged at Bristol Crown court of conspiring to steal money by altering cards in this way. They produced an eminent banking industry expert who testified that what they had planned was impossible; but after a leading newspaper demonstrated otherwise, they changed their plea to guilty [8].
- Some banks have schemes that enable PINs to be checked by off-line ATMs without giving them the master encryption key needed to derive PINs from account numbers. For example, customers of one British bank got a credit-card PIN with digit-one plus digit-four equal to digit-two plus digit-three, and a debit-card PIN with one plus three equals two plus four. Villains eventually discovered that they could use stolen cards in off-line devices by entering a PIN such as 4455.
- Even without such weaknesses, the use of store-and-forward processing is problematic. Anyone can open an account, get a card and PIN, make several copies of the card, and get accomplices to draw cash from a number of different ATMs at the same time. This was a favorite modus operandi in Britain in the mid-1980s, and is still a problem in Italy, where ATMs are generally off-line over the weekend.
- Any security technology can be defeated by gross negligence. In August 1993, my wife went into a branch of our bank and told them that she had forgotten her PIN; they helpfully printed a replacement PIN mailer from a PC behind the counter. This was not the branch at which her account is kept; no one knew her, and the only identification she produced was her bank card and checkbook. By that time, banks in Britain had endured some 18 months of bad publicity about poor ATM security, and this particular bank had been a press target since April of that year.

This might lead us to ask what the future might hold. Will all magnetic cards be replaced with smartcards, as is already happening in countries from France to Guatemala and from Norway to South Africa [2]? One of the smartcard vendors' strongest arguments is that card forgery keeps on rising, and that the fastest 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; more recently, an enterprising gang bought ATMs and an ATM software development kit (on credit), programmed a machine to capture PINs, and rented space for it in a shopping mall in Connecticut. A Dutch gas station attendant used a tapped point-of-sale terminal to harvest card data in 1993; and in March 1994, villains constructed an entire bogus bank branch in the East End of London and made off with £250,000 ($375,000). There seems to be no defense against this kind of attack, short of moving from magnetic cards to payment tokens, which are more difficult to forge.

But trusting technology too much can be dangerous. Norwegian banks spent millions on smartcards, and are now as publicly certain about their computer security as their British colleagues. Yet despite the huge investment, there have been a number of cases in Trondheim, Norway, where stolen cards have been used without the PIN having been leaked by the user. The banks' refusal to pay up will probably lead to litigation, as in Britain, with the same risk to both balance sheets and reputations.

Where transaction processing systems are used directly by the public, there are really two separate issues. The first is the public-interest issue of whether the burden of proof (and thus the risk) falls on the customer or on the system operator. If the customer carries the risk, the operator will have little short-term incentive to improve security; but in the longer term, when innocent people are prosecuted because of disputed transactions, the public interest becomes acute.

If, on the other hand, the system operator carries the risk, as in the United States, then the public-interest issue disappears, and security becomes a straightforward engineering problem for the bank (and its insurers and equipment suppliers). We consider how this problem can be tackled in the following sections.

Organizational Aspects

First, a caveat: our research showed that the organizational problems of building and managing secure systems are so severe that they will frustrate any purely technical solution.

Many organizations have no computer security team at all, and the rest have tenuous arrangements. The internal audit department, for example, will resist being given any line management tasks, while the programming staff dislike anyone whose role seems to be making their job more difficult. Security teams thus tend to be "reorganized" regularly, leading to a loss of continuity; a recent study shows, for example, that the average job tenure of computer security managers in U.S. government departments is only seven months [13].

It should not be surprising that many firms get outside consultants to do their security policy-making and review tasks. However, this can be dangerous, especially if firms pick these suppliers for an "air of certainty and quality" rather than for their technical credentials. For example, there was a network of over 40 banks in Asia that encrypted their PINs in a completely insecure manner (using a Caesar cipher) for five years, yet in all this time not one of their auditors or consultants raised the alarm. It is interesting to note that, following a wave of litigation, accountancy firms are rewriting their audit contracts to shift all responsibility for fraud control to their clients; but it remains to be seen what effect this will have on their security consulting business.

Much of the management debate, however, is not about the consultancy

or facilities management issue but whether information security should be centralized or not. A useful parallel here may be to compare train and aircraft systems. Railways keep tight central control; if the train driver falls asleep, or goes through a red light, the train stops automatically. In civil aviation, on the other hand, the pilot remains in command, and progress has not deskilled the job but made it ever more complex and demanding.

Both the railway and airline models find reflections in current security practice and research, but the railway model is dominant, due partly to the historical dominance of mainframes, and partly to the influence of government secure computing standards, such as the U.S. Orange Book [23].

Some authors argue that the shift to client-server architectures will force security systems to be decentralized [8]. It may not be wise, however, to take this to extremes. In many organizations, centralization is a cyclic phenomenon; every so often, power is devolved from the center in order to create vigorous, autonomous business units, and profits rise for a while. Eventually, directors become worried that the business units are going in different directions, or there may even be a disaster in a maverick subsidiary; either way, the head office decides to reassert its authority [19].

This can cause problems. Whether an organization has a centralized security team or devolves the responsibility to line management, it can take many years for a security capability to mature and become effective. Continuity matters; and we do not really understand how to maintain effective control in an organization whose structure is constantly changing.

The Problems with Security Products

We found that almost all attacks on banking systems involved blunders, insider involvement, or both. High-tech attacks were rare, and the two that did occur were possible because in one case PINs were sent in an obvious manner, and in the other the authorization responses were; so these can be seen as the effect of the absence of security rather than some particular problem with it.

However, high-tech threats were the ones that most exercised the cryptographic equipment industry, and the ones that their products were explicitly designed to prevent. But these products are often so difficult to integrate into larger systems that they can contribute significantly to the blunders that caused the actual losses.

The same turns out to apply to classified systems, too. A recent U.S. Air Force survey revealed that poor implementation is their main security problem: although a number of systems employ "trusted" components, there are few, if any, fielded systems that use them effectively. In fact, they often had a negative effect, as they fostered complacency; and the assumption that programmers would be careless was self-fulfilling. The National Security Agency has also recently admitted that most security failures in its area of interest occur at the level of implementation detail [22]. Thus, although the gory details remain classified, it is clear that our findings are just as relevant to military systems as to banking networks.

This should worry policymakers, as government standards such as the Orange Book have directed vast sums of both public and private investment toward developing the pool of "trusted" products. Our results suggest this effort has been misguided, and has misled people into neglecting the more important problem of how cryptographic and other security features are embedded in real systems.

Equipment vendors may argue that skill in cryptology is rare, being restricted to universities, government laboratories, and their own design labs; and that this skill shortage ensures that cryptographic know-how will have to be brought to market in the form of products. There may be some truth in this point of view, and many companies and government departments will in any case buy whatever products are recommended by the appropriate authority. However, because they lack skills at security integration and management, they will go on to build systems with "holes."

This is a failure of the certification process. One would not think highly of an inspector who certified the Boeing 747 or the Sukhoi Su-26 for use as a basic trainer, as these aircraft take a fair amount of skill to fly. The aviation community understands this, and formalizes it through a hierarchy of licenses—from the private pilot's license for beginners, through various commercial grades, to the airline license, which is a legal requirement for the captain of any scheduled passenger flight.

The world of computer security has not yet caught up. Given that most managers and staff cannot be assumed to have any specialized knowledge at all, security products should only be certified if they are simple enough for ordinary technical staff to use.

Aircraft engineers are also aware that accidents usually have multiple causes, while security researchers tend to use threat and risk models in which only one thing goes wrong at a time. Yet in the large majority of ATM frauds, the cause was a combination: carelessness by insiders plus an opportunist attack by outsiders (or by other insiders).

How can we cope with the reality of unskilled implementers and multiple threats? One line of attack is to try to make security systems robust. This might mean that they can tolerate minor errors in system design, implementation, and operation with no loss of security, or with—at worst—a graceful degradation; alternatively, we might try to design the components so they cannot be fitted together in unsafe ways. Robustness may be the most important problem in security; yet there has been very little research done on this topic.

The Nature of Robustness

Robustness principles are familiar enough in other engineering disciplines: their essence is to prefer a solution that may be slightly less than optimal, but that provides some benefit such as reducing design or operational complexity, facilitating analysis, providing resilience against minor errors in design and operation, and providing redundancy against component failure.

However, the exact flavor of robustness varies from one engineering discipline to another. Bridge builders

know that most likely faults, such as a poor batch of steel, contaminated concrete, or uneven weathering, have the effect of slightly reducing the breaking strain of the structure; so the usual rule is to design a bridge so that its theoretical breaking strain with optimal materials is six times what is required, and to proof test samples of the actual materials used to three times the needed strength.

Aircraft engineers, on the other hand, know that many accidents are caused by the failure of critical components, and make extensive use of redundancy; with very critical functions, this may extend to design diversity. When flying in clouds, pilots need to know which way is up, and so a modern airliner typically has two attitude indicators driven by electrically powered gyro platforms. If these both fail at once, there is a 1950s-era technology artificial horizon with pneumatically driven gyros, and a 1920s vintage turn-and-slip indicator driven by a battery.

But neither overdesign nor redundancy is adequate for secure computational systems. Just doing more rounds of a bad encryption algorithm, or using a number of weak algorithms one after another, will not necessarily produce a strong one; and unthinking use of redundancy in computer systems can be dangerous, as resilience can mask faults that would otherwise be found and fixed.

Our work on ATM systems therefore inspired us to look for an organizing principle for robustness properties in computer security systems. The key insights came from the high-tech end of the business—from studying authentication protocols and the ways in which cryptographic algorithms interact (see the sidebar "No Silver Bullet").

These results suggest that explicitness should be the organizing principle for security robustness. Cryptographic algorithms interact in ways that break security when their designers do not specify the required properties explicitly; and protocol failures occur because naming, freshness, and chaining properties are assumed implicitly to hold between two parties.

The importance of explicitness is confirmed in the field of operating systems security by a recent report that shows implicit information problems were one of the main causes of failure there, and that most of the others were due to obvious requirements not being explicitly checked [17].

However, just saying that every security property must be made explicit is not a solution to the practical problem of building robust systems. The more aspects of any system are made explicit, the more information its designer has to deal with; and this applies not only to designing systems, but to evaluating them as well. Can our explicitness principle ever amount to more than a warning that all a system's assumptions must be examined very carefully?

There are two possible ways forward. The first is to look for ways in which a system that has a certain set of relationships checked explicitly can be shown using formal methods to possess some desirable security property. This may be a good way to deal with compact subsystems such as authentication protocols; and lists of the relevant relationships have been proposed [1].

The other, and more general, approach is to try to integrate security with software engineering. Data-dependency analysis is already starting to be used in the security world:

- A typical difficult problem is identifying which objects in a system have security significance. As we saw previously, frauds have taken place because banks failed to realize that an address change was a security event; and evaluating the significance of all the objects in a distributed operating system is a Herculean task, which involves tracing dependencies explicitly. Automated tools are now being constructed to do this [11];
- Another difficult problem is that of verifying whether an authentication protocol is correct. This problem can be tackled by formal methods; the best-known technique involves tracing an object's dependencies on crypto keys and freshness information [9], and has been used to verify transaction processing applications as well [2].

However, we cannot expect to find a "silver bullet" here either. This is because many of the more subtle and difficult mistakes occur where assumptions about security properties fail at the interface between different levels (e.g., algorithm-protocol or protocol-operating system) [6]. Thus when we decompose our system into modules, we must be very careful to ensure that all our assumptions about possible interactions have been made explicit and considered carefully.

Explicitness and Software Engineering

Robustness as explicitness fits in well with the general principles of software engineering but may require some changes in its practice. A recent study shows that for many years the techniques used by system builders to manage security requirements, assumptions, and dependencies have lagged a generation behind the state of the art [5].

An even more elementary problem concerns the mechanisms by which security goals are established. Many software engineering methodologies since the waterfall model have dispensed with the traditional requirement that a plain-language "concept of operations" should be agreed upon before any detailed specification work is undertaken. This is illustrated by our work on ATMs.

ATM security involves several conflicting goals, including controlling internal and external fraud, and arbitrating disputes fairly. This was not understood in the 1970s; people built systems with the security technology they had available, rather than from any clear idea of what they were trying to do. In some countries they ignored the need for arbitration altogether, with expensive consequences. This underlines the particular importance of making security goals explicit, where a concept of operations can be a great help; it might have focused ATM designers' attention on the practicalities of dispute resolution.

Finally, it may be helpful to compare secure systems with safety critical systems. They are known to be related: while the former must do at *most* X, the latter must do at *least* X, and there is a growing realization that many of the techniques and even

components from one discipline can be re-used in the other. Let us extend this relationship to the methodological level and have a look at good design practice. A leading software safety expert has summed this up in four principles [20]:

- The specification should list all possible failure modes of the system. This should include every substantially new accident or incident that has ever been reported and that is relevant to the equipment being specified.
- It should explain what strategy has been adopted to prevent each of these failure modes, or at least make them acceptably unlikely.
- It should then spell out how each of these strategies is implemented, including the consequences when each single component fails. This explanation must cover not only technical factors, but training and management issues too. If the procedure when an engine fails is to continue flying with the other engine, then what skills does a pilot need to do this, and what are the procedures whereby these skills are acquired, kept current, and tested?
- The certification program must include a review by independent experts, and test whether the equipment can in fact be operated by people with the stated level of skill and experience. It must also include a monitoring program whereby all incidents are reported to both the equipment manufacturer and the certification body.

This structure ties in neatly with our findings, and gives us a practical paradigm for producing a robust, explicit security design in a real project. It also shows that the TCSEC program has a long way to go. As we mentioned earlier, so far no one seems to have attempted even the first stage of the safety engineering process for cryptographic systems. We hope that this article will contribute to closing the gap, and to bringing security engineering up to the standards already achieved by the safety-critical systems community.

Conclusions

Designers of cryptographic systems have suffered from a lack of feedback about how their products fail in practice, as opposed to how they might fail in theory. This has led to a false threat model being accepted; designers focused on what could possibly go wrong, rather than on what was likely to, and many of their products ended up being so complex and tricky to use, they caused implementation blunders that led to security failures.

Almost all security failures are in fact due to implementation and management errors. One specific consequence has been a spate of ATM fraud, which has not only caused financial losses, but has also caused several wrongful prosecutions and at least one miscarriage of justice. There have also been military consequences, which have now been admitted (although the details remain classified).

Our work also shows that component-level certification, as embodied in the TCSEC program, is unlikely to achieve its stated goals. This, too, has been admitted indirectly by the military; and we would recommend that future security standards take much more account of the environments in which the components are to be used, and especially the system and human factors.

Most interesting of all, however, is the lesson that the bulk of the computer security research and development budget is expended on activities that are of marginal relevance to real needs. The real problem is how to build robust security systems, and a number of recent research ideas are providing insights into how this can

No Silver Bullet

There has been a surge of recent interest in building robust authentication protocols. These protocols are short sequences of messages that are used for automated crypto key distribution in distributed systems, and are surprisingly hard to get right (in the sense that there are no possible message modification attacks). Mistakes have been found in some of them more than a decade after they were first published [1].

During 1993, three papers were published in which the authors independently proposed robustness as a solution. The basic idea is that since attacks on protocols depend on missing information, one could prevent them by including everything relevant, including the sender and recipient of each message [12], the context [6], and a hash of the previous message step [25].

Each of these proposals only addresses part of the problem, and none of them is adequate on its own: protocol failures are known which result from the lack of name, or of freshness, or of context information within the security envelope [1]. But putting them together and insisting on all these variables being made explicit in each message appears to solve the global robustness problem—at least for simple protocols.

This combined approach had actually been adopted in 1991 for a banking application [2], in which attacks on the payment protocols are prevented by making each message start with the sender's name, and then encrypting it under a key that contains a hash of the previous message. These techniques were not tried as an experiment in robustness, but to facilitate formal verification.

Another reason to believe that explicitness should be the organizing principle for robust security comes from studying how cryptographic algorithms interact. Researchers have asked, for example, what sort of properties we need from a hash function in order to use it with a given signature scheme, and a number of necessary conditions have been found. This led us to ask whether there is any single property that is sufficient to prevent all dangerous interactions. We recently showed that the answer is probably no [4].

What this means is that in cryptology, as in software engineering, we cannot expect to find a "silver bullet" [7]; there can be no general property that prevents two algorithms from interacting and that is likely to be of any practical use. In most real situations, however, we can explicitly specify the properties we need; typical properties might be that a function f is correlation-free (we can't find x and y such that $f(x)$ and $f(y)$ agree in too many bits) or multiplication-free (we can't find x, y, and z such that $f(x)f(y) = f(z)$).

be achieved. The fundamental organizing principle for security robustness properties appears to be explicitness.

Robust security designs are those that make their assumptions explicit, and so the design methodology must force the team to examine its assumptions in a systematic and careful manner. This has been shown to be central in the design of cryptographic algorithms, authentication protocols, and secure operating systems; it is no less important at the application level, where the relevant explicitness properties appear to fit well with experience accumulated in the safety-critical systems community.

Above all, we must be explicit about what security goals we are trying to achieve. Security is not a simple Boolean predicate; it concerns how well a system performs certain functions. Indeed, there is a sense in which there are no "secure" systems at all; there are merely computer systems whose goals include beating foreign armies, preventing fraud, or winning lawsuits. If these goals are not made explicit, they are unlikely to be achieved. ■

References

1. Abadí, M., and Needham, R.M. Prudent Engineering Practice for Cryptographic Protocols. Tech. Rep. 125, DEC SRC, June 1994.
2. Anderson, R.J. UEPS—A Second Generation Electronic Wallet. In *Computer Security—ESORICS 92*. Lecture Notes in Computer Science, vol. 648. Springer-Verlag, New York, pp. 411–418.
3. Anderson, R.J. Why cryptosystems fail. In *Proceedings of the 1993 ACM Conference in Computer and Communications Security*, pp. 215–227.
4. Anderson, R.J. The classification of hash functions. In *Proceedings of the 4th IMA Conference in Cryptography and Coding (1993)*. To be published.
5. Baskerville, R. Information systems security design methods: implications for information systems development. *ACM Computing Surveys 25*, 4 (Dec. 1993), 375–414.
6. Boyd, C., and Mao, W.B. Limitations of logical analysis of cryptographic protocols. In *Pre-Proceedings of Eurocrypt 93*, pp. T88–T96.
7. Brooks, F.P. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, Reading, Mass., 1975.
8. Bull, J.A., Gong, L., and Sollins, K. Towards security in an open systems federation. In *Proceedings of ESORICS 92*. Lecture Notes in Computer Science, vol. 648. Springer-Verlag, New York, pp. 3–20.
9. Burrows, M., Abadi, M., and Needham, R.M. A logic of authentication. In *Proceedings of the Royal Society of London A*, vol. 426, 1989, pp. 233–271.
10. Butler, R.W., and Finelli, G.B. The infeasibility of experimental quantification of life-critical software reliability. In *Proceedings of the ACM Symposium on Software for Critical Systems*, New Orleans, La., Dec. 1991, pp. 66–76.
11. Faigin, D.P., Donndelinger, J.J., and Jones, J.R. A rigorous approach to determining objects. In *Proceedings of the 9th Annual Computer Security Applications Conference*, IEEE, 1993, pp. 159–168.
12. Gong, L. Thoughts on cryptographic protocols. In *Proceedings of the 1993 Cambridge Protocols Workshop*. Lecture Notes in Computer Science. Springer-Verlag, New York. To be published.
13. Highland, H.J. Perspectives in information technology security. In *Proceedings of the 1992 IFIP Congress, 'Education and Society'*, IFIP A-13 vol. 2, 1992, pp. 440–446.
14. Jack, R.B. (chairman), *Banking services: law and practice report by the Review Committee*, HMSO, London, 1989.
15. *Dorothy Judd v Citibank*, in 435 NYS, 2d series, pp. 210–212, 107 Misc.2d 526.
16. Kahn, D. *The Codebreakers*. Macmillan, New York, 1967.
17. Landwehr, C.E., Bull, A.R., McDermott, J.P., and Choi, W.S. A taxonomy of computer program security flaws, with examples. U.S. Naval Research Laboratory report NRL/FR/5542–93–9591.
18. Lewis, B. How to rob a bank the cash-card way, *The Sunday Telegraph*, 25 April 1993, p. 5.
19. Macrae, N. Sir Humphrey fudges his half-reforms, *The Sunday Times* 17 July 1994, sec. 4, p. 4.
20. McDermid, J.A. Issues in the development of safety critical systems. Public Lecture, Cambridge, 3 Feb. 1993.
21. Meyer, C.H., and Matyas, S.M. *Cryptography: A New Dimension in Computer Data Security*, John Wiley & Sons, New York, 1982.
22. Morris, R. In *Proceedings of the 1993 Cambridge Protocols Workshop*. Lecture Notes in Computer Science. Springer-Verlag, New York. To be published.
23. U.S. Department of Defense, *Trusted Computer System Evaluation Criteria*, 5200.28-STD, December 1985.
24. Wiener, M.J. *Efficient DES Key Search*, Technical report TR-244, School of Computer Science, Carleton University, Ottawa, May 1994.
25. Woo, T.Y.C., and Lam, S.S. A semantic model for authentication protocols. In *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 178–194.

About the Author:

ROSS J. ANDERSON is editor of *Computer and Communications Security Reviews*; he has worked on cryptology and computer security for the last 10 years, and consulted for a wide range of equipment manufacturers and users. Current research interests focus on the performance and reliability of computer security systems. **Author's Present Address:** Cambridge University Computer Laboratory, Pembroke Street, Cambridge CB2 3QG, United Kingdom, rja14@cl.cam.ac.uk

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© ACM 0002-0/82/94/1100 $3.50

An earlier version of this article appeared in the *Proceedings of the 1993 ACM Conference on Computer and Communications Security*. For reasons of space, it has been reduced to about half the original length, and many of the case histories, references, and technical details have been omitted. Readers are referred to the original paper [3] for this information.