

2

Protocol Analysis, Composability and Computation

Ross Anderson, Michael Bond

Security protocols—early days

The study of security protocols has been associated with Roger Needham since 1978, when he published the seminal paper on the subject with Mike Schroeder [2].

The problem they investigated was how to distribute cryptographic keys in a network of computers. One solution is to have an authentication service with which all the principals share a key; then if Alice wants to chat with Bob (for example) she can call the service and get two encrypted messages containing the same session key—one encrypted under the key she shares with the service so she can read it, and one encrypted under the key Bob shares with the service so Bob can read it. She can now send the second of these to Bob to establish secure communication. The mechanism that Needham and Schroeder designed for this evolved into Kerberos, which is now part of Windows and is probably the most widely used of all authentication protocols.

Security protocols are now embedded in a great many applications, but it is common to find unexpected bugs in them. For example, many banks used to encrypt each customer's PIN using a key known to their ATMs and write it on the ATM card magnetic strip. The idea was to provide a limited service when the network was down. Years later, a villain discovered that the account number and the encrypted PIN were not linked: he could make up a bank card with his own encrypted PIN but someone else's account number, and loot their account. He went on to steal a lot of money, and once in prison wrote a manual telling everyone else how to do it too. The banks had to spend millions on changing their systems.

Clarifying the assumptions

Researchers started to gnaw away at the protocols described in the literature and found fault with essentially all of them. The failure to bind protocol elements was one frequent problem; another was that old messages could be replayed. In the case of the original Needham-Schroeder protocol, for example, the freshness of the key generated by the server was guaranteed to only one of the principals. This was not necessarily an attack, as its inventors only claimed to protect honest insiders from dishonest outsiders. However, it led to a debate about the assumptions underlying security protocol design. Do we protect only against outsiders, or against insiders? Against the malicious, or the merely careless? For example, if we use timestamps to guarantee protocol freshness, are we vulnerable to principals who carelessly let their clocks run slow? Do we only consider an attacker to have won if he can impersonate an authorised principal, or do we need to stop people abusing the protocol mechanisms to perform a service denial attack?

The early attacks led to a second seminal paper, which Roger wrote with Mike Burrows and Martin Abadi in 1989 [1], and which introduced a logic of authentication. This enables an analyst to formalise the assumptions and goals of a security protocol, and to attempt to prove its correctness. When a proof cannot be found, the place at which one gets stuck often shows where an attack can be mounted. This style of analysis turned out to be very powerful, and a large literature quickly developed in which the “BAN Logic” and other formal tools were developed and extended to tackle a range of problems in protocol design.

One of the remarkable things about the study of security protocols is that they have not become a solved problem. One might think that managing the objects associated with authenticating users over a network—passwords, keys and the like—was a fairly compact problem which would have been done to death within a few years. However, the more we dig, the more we find.

Between 1992 and 2002, Roger hosted a protocols workshop every Easter. Early events dwelt on matters of authentication and logic, but by the mid-90s, the growing interest in electronic commerce was yielding papers on mechanisms for micropayments, bets, streaming media, mobile communications and electronic voting. Later years brought work on PKI, trust management and copyright enforcement. More and more problems come along as more and more businesses reinvent themselves online; threat models have also become more realistic, with dishonest insiders displacing the mythical ‘evil hacker on the Internet’.

Dishonest insiders, and the composition problem

Over the last two years, we have been exploring exactly how one might re-engineer cryptography to cope with dishonest insiders. One conclusion is that the analysis of security protocols must be extended to application programming interfaces. This is because the crypto keys used in authentication and payment pro-

protocols are often kept in separate hardware security processors, or at least in cryptographic libraries, to which access can be restricted using physical or logical mechanisms. However, an interface has to be exposed to the application program, which will occasionally be suborned—whether by a corrupt insider, or by malware. How much harm can be done, and how can we limit it?

Protecting protocols was hard enough, and yet the typical protocol consists of 3–5 messages exposed to manipulation. The API of a modern crypto library or hardware cryptoprocessor may contain 30–500 callable functions, many with a range of options. This provides a very rich and complex environment for mischief.

Attacks often involve using two separate mechanisms provided by the cryptoprocessor for different purposes, each of which could be innocuous by itself but which combine to cause trouble. For example, it is common to compute a customer PIN by encrypting the account number with a ‘PIN derivation key’: the cryptoprocessor then returns the PIN encrypted with a PIN storage key, so that the application has no access to its clear value. So far, so good. Then there is another transaction that can be used to encrypt a communications key under the terminal key loaded in an ATM. Here things start to go wrong, as the cryptoprocessor does not distinguish between a terminal key and a PIN derivation key; it considers them both to be of the same type. The upshot is that an attacker can supply the device with an account number, claiming that it is a communications key, and ask for it to be encrypted under the PIN derivation key.

Attacks like this extend protocol analysis all the way to the composition problem—the problem that connecting two systems that are secure in isolation can give a composite system that leaks. This had previously been seen as a separate issue, tackled with different conceptual tools.

Differential protocol analysis

We are now working on the second generation of API attacks, which exploit the application syntax supported by the cryptographic service. These attacks are even more powerful, and at least as interesting from the scientific point of view. PIN generation provides a neat example here too. In more detail, the standard PIN computation involves writing the result of the encryption as a hex string and decimalising it. As some banks like to let customers change their PIN to a more memorable number, there is a provision to add an offset to give the PIN that the customer actually enters:

Account number:	8807 0123 4569 1715
PIN derivation key:	FEFE FEFE FEFE FEFE
Encrypted account number:	A2CE 126C 69AE C82D
Natural (decimalised) PIN:	0224
Offset:	6565
Customer PIN:	6789

The typical implementation requires the programmer to send the cryptoprocessor the account number, a table describing the decimalisation (here, ‘0123 4567 8901 2345’) and the offset. The processor returns the PIN, encrypted under the PIN storage key.

The designers do not seem to have realised that a crooked programmer can manipulate the decimalisation table and the offset as well as the account number. A multitude of attacks follow. For example, one can send in an account number with a decimalisation table of ‘1111 . . . 11’ to find out the ciphertext corresponding to a clear PIN of ‘1111,’ and then with a decimalisation table of ‘0111 . . . 11’ to see if there is a zero in the first four digits of the encrypted account number (if so, the PIN, and thus the ciphertext output, will be different). By manipulating the decimalisation table further, he can get all the digits in the PIN, and by then playing with the offset he can get their order. In total, the attack requires only 15–25 unprivileged cryptoprocessor transactions to discover the PIN on a single target account.

This second type of attack takes protocol analysis into yet another realm: that of differential attacks. Over the last ten years, a number of techniques have been invented for attacking cryptographic systems by bombarding them with inputs with chosen differences. For example, in differential cryptanalysis, one analyses the changes in the output of the encryption algorithm; while with differential power analysis, one measures changes in the current consumption or electromagnetic emissions of the equipment. Now we have examples of how consecutive runs of a protocol can leak information if the inputs are suitably chosen. The resulting ‘differential protocol analysis’ appears to be very powerful against application-level crypto.

It will take us some time to figure out the general lessons to be drawn from attacks like this, the robustness principles that designers should use to avoid them, and the analysis techniques that might assure us of a particular design’s soundness. The randomisation of all protocols (another feature of Roger’s work) is likely to be important.

Quantitative analysis and multiparty computation

Various researchers have speculated about whether there might one day be a quantitative analysis of protocol security. This might be feasible for PIN processing applications as we can measure the information leakage per transaction in terms of the reduction of entropy in the unknown PIN. This leads in turn to a possible real-world application of an attack previously considered theoretical.

Gus Simmons wrote extensively on covert channels in protocols. One such channel that is always present is the ‘balking channel’—when one of the principals in a protocol signals something by halting and refusing to continue. This is normally considered unimportant as its information capacity is only a third of a bit per transaction. But with systems designed to cope with large transaction vol-

umes, this need no longer hold. For example, a Trojanned cryptoprocessor could balk when it sees a predetermined PIN. If the PIN length were eight digits, this would be unlikely to hinder normal operation, but at a thousand transactions a second, a programmer could quickly find a number in a typical nine-digit account number range with just this PIN, and open an account for it. Once this kind of problem is appreciated, one can start to look for attacks that involve inducing rare error conditions that cause the cryptoprocessor to abort a transaction. (They exist.)

A third emerging link is between protocol analysis and secure multiparty computation. In application-level crypto we may have several inputs to a computation, some of them coming from an untrusted source, and we have to stop users manipulating the computation to get outputs useful for bad purposes. In the PIN decimalisation example above, one might try to solve the problem by blocking tables such as ‘1111...11.’ Yet an attacker can get by with scarcely more work by using two normal-looking tables that differ slightly (another kind of differential attack). We might therefore think that if we can’t sanitize the inputs to the computation, perhaps we can authenticate them, and use only those tables that real banks actually use. But building every bank in the world into our trust base is what we were trying to avoid by using cryptography!

Conclusion

The protocol work that started off a quarter of a century ago may have seemed at the time like a minor detail within the larger project of designing robust distributed systems. Yet it has already grown into the main unifying theme of security engineering. Application-level protocols, and especially those from which an attacker can harvest data over many runs, open up new problems. The resulting analysis techniques are set to invade the world of composable security, and the world of multiparty computation. The influence, and the consequences, of Roger’s contribution just keep on growing.

References

- [1] BURROWS, M. ABADI, M. AND NEEDHAM, R.M., ‘A logic of authentication,’ *ACM Trans. on Computer Systems*, vol. 8, no. 1, pp. 18-36, 1990.
- [2] NEEDHAM, R.M. AND SCHROEDER, R.M., ‘Using encryption for authentication in large networks of computers.’ *Comm. ACM*, vol. 21, no. 12, pp. 993-999, 1978.

