Computer Science Tripos part 2 – Security

Lecture 11 – Public Key Protocols

Ross J. Anderson

1 Introduction

We saw that the protocols in which symmetric cipher systems are used, such as Needham Schroder and the Wide Mouthed Frog, are surprisingly easy to get wrong, and the same applies to protocols that use public key algorithms. The main difficulties with public key protocols are:

- Firstly, we need some mechanism to bind users to keys. This might involve a third party in certifying public keys, or in running a secure directory service. Arranging for compromised keys to be revoked in a timely manner can be expensive; and there is a lot of politicking going on about control of the future 'public key infrastructure'.
- As we discussed in the lecture on conventional cryptographic protocols, authenticity = integrity + freshness. Thus even if the public key scheme assures the integrity of a message, we will often want the recipient to be able to check that a message is recent, rather than a replay.
- Public key algorithms have mathematical properties, such as being multiplicative, which can interact with the protocol design in novel and unpleasant ways.

An actual protocol might be something like

$$A \rightarrow B : CA, \{\{T_A, B, M\}_{K_A^{-1}}\}_{K_B} \qquad (1)$$

Here CA is a certificate of Alice's public key, which is signed by our trusted third party Sam; T_A is a timestamp; and B is Bob's name. We will assume that there is enough redundancy in the message and the namespace that no-one can find a block by random trial which encrypts under Alice's public key to a string which looks like a valid $\{T_A, B, M\}$.

The public key certificate CA might have the form:

$$CA = \{T_S, L, A, K_A\}_{K_S^{-1}} \qquad (2)$$

Here, T_S is the certificate's starting date and time, and L is the length of time for which it is valid.

1.1 Protocol failure 1 – Denning-Sacco

There is another reason to put B's name inside the security envelope in protocol (1) above, other than just providing enough redundancy to stop a forward search attack by Charlie.

One of the first proposed public key protocols was due to Denning and Sacco, who in 1982 pointed out the revocation flaw of the Needham Schroder protocol and proposed instead that conventional encryption keys should be distributed using a public key mechanism. Their proposal was as follows, where CX is a public key certificate for X of the form in (2) above:

Message 1	$A \rightarrow S:$	A, B
Message 2	$S \rightarrow A:$	CA, CB
Message 3	$A \rightarrow B:$	$CA, CB, \{\{T_A, K_{AB}\}_{K_A^{-1}}\}_{K_B}$

In 1994, Abadi and Needham pointed out that this protocol is fatally flawed. Bob, on receiving this message, can masquerade as Alice for as long as Alice's timestamp T_A remains valid!

To see how, suppose that Bob wants to masquerade as Alice to Charlie. He goes to Sam and gets a fresh certificate CC for Charlie, and then strips off the outer encryption $\{...\}_{K_B}$ from message 3 in the above protocol. He now re-encrypts the signed key $\{T_A, K_{AB}\}_{K_A^{-1}}$ with Charlie's public key — which he gets from CC — and makes up a bogus message 3:

Message 3	$B \rightarrow C:$	$CA, CC, \{\{T_A, K_{AB}\}_{K_A^{-1}}\}_{K_C}$

1.2 Protocol failure 2 – X.509

Another problem was found by Burrows, Abadi and Needham in the CCITT X.509 protocol. This is described in some detail their paper which is appended to the notes for lecture 4; briefly, the idea is that Alice may sign a message of the form $\{T_A, N_A, B, X, \{Y\}_{K_B}\}$ and send it to Bob. Here T_A is a timestamp, N_A is a serial number, and X and Y are user data. The problem here is that since the order of encryption and signature are reversed — Y is first encrypted, then signed — it is possible for Charlie to strip off Alice's signature and add one of his own.

In 1995, we found an even sneakier attack. With many public key algorithms it is possible, given a message M and a ciphertext C, to find a public key K such that $\{M\}_K = C$ (because public key encryption is not in general collision-free). Consider for example RSA with a 512 bit modulus. If Alice sends Bob M, and if the modulus, public exponent and private exponent of party α are n_α, e_α and d_α, and if we ignore hashing (which makes no difference), the signed encrypted message would be $\{M^{e_B} \pmod{n_B}\}^{d_A} \pmod{n_A}$. But since Bob can factor n_B and its factors are only about 256 bits long, he can work out discrete logarithms with respect to them and then use the Chinese Remainder

Theorem to get discrete logs modulo n_B. So he can get Alice's 'signature' on M' by finding x such that $[M']^x = M \pmod{n_B}$ and then registering (xe_B, n_B) as a public key. The effect is that, in the above example, if Bob wants to convince a judge that Alice actually sent him the message Z rather than Y, he can decipher $\{Y\}_{K_B}$, find another key K' such that $\{Z\}_{K'} = \{Y\}_{K_B}$, and register this new key K' with a certification authority.

1.3 Protocol failure 3 – TMN

Yet another problem is that most public key algorithms have some mathematical properties which an opponent might exploit. For this reason, we cannot treat public key algorithms as 'black boxes' – we have to look under the hood and see how they work (hence the number theory and factoring).

At Crypto 89, Tatebayashi, Matsuzaki and Newmann proposed a cryptographic protocol whose purpose was to have Sam do most of the computational work in setting up a key between Alice and Bob (note that calculating r^3 modulo a 1000 bit number takes two modular multiplications, while working out r^d will take about 1500). The idea was that Alice and Bob would be using smartcards in mobile phones to do their cryptography.

In this protocol, Sam has an RSA modulus n, and both Alice and Bob send him a random number (respectively r_A and r_B) which they have cubed modulo n. Sam extracts the cube roots, sends Alice $r_A \oplus r_B$, and now Alice and Bob can communicate using r_B as their encryption key:

Message 1	$A \to S:$	$A, B, r_A^3 \pmod{n}$
Message 2	$S \to B:$	A, B
Message 3	$B \to S:$	$r_B^3 \pmod{n}$
Message 4	$S \to A:$	$r_A \oplus r_B$

However, this is not secure, as Sam is allowing himself to be used as an oracle. If Charlie wants to recover r_B – which Alice and Bob will by now be using as a session key – then all he has to do is run the protocol with a buddy, David. Charlie observed r_B from Alice and Bob's run of the protocol, and he now chooses a random r_C and sends $r_B^3 r_c^3$ to Sam. David sends r_D^3, where r_D is a value known to Charlie and prearranged with him. The protocol now runs as follows:

Message 1	$C \to S:$	$C, D, r_B^3 r_C^3 \pmod{n}$
Message 2	$S \to D:$	C, D
Message 3	$B \to S:$	$r_D^3 \pmod{n}$
Message 4	$S \to A:$	$r_B r_C \oplus r_D$

Charlie xor's thus value with r_D and divides by r_C to get R_B.

The basic problem here (and with several other bad protocols) is that Sam allows himself to be used as an oracle by Charlie.

1.4 Protocol failure 4 – Needham Schroder

Needham and Schroder also proposed an authenticated key exchange protocol that runs as follows. Here we assume for the sake of brevity that Alice and Bob both already have a copy of the other's key certificate.

$$A \longrightarrow B : \{N_A, A\}_{K_B}$$
$$B \longrightarrow A : \{N_A, N_B\}_{K_A}$$
$$A \longrightarrow B : \{N_B\}_{K_B}$$

However, Lowe pointed out a middleperson attack. Here, Charlie sits between Alice and Bob. He appears as Charlie to Alice, but pretends to be Alice to Bob:

$$A \longrightarrow C : \{N_A, A\}_{K_C}$$
$$C \longrightarrow B : \{N_A, A\}_{K_B}$$
$$B \longrightarrow C : \{N_A, N_B\}_{K_A}$$
$$C \longrightarrow A : \{N_A, N_B\}_{K_A}$$
$$A \longrightarrow C : \{N_B\}_{K_C}$$
$$C \longrightarrow B : \{N_B\}_{K_B}$$

1.5 Protocol failure 4 – Chosen protocol attack

Some governments and smartcard vendors have the idea that a citizen's public-private keypair could be used for many different purposes, and significant amounts of money have been invested in developing 'multi-application smartcards'. This has met with resistance from businesses for practical reasons (whose logo appears on the card? who is liable when things go wrong? who controls the mailing list?) – but there is also a technical problem.

Suppose that a payment protocol involves the user signing the hash of a message containing the same kind of information you find in a cheque. An attacker might design a protocol for another purpose, such as declaring that the user is over 18 and entitled to access a porn site, and arrange that the user has to sign a challenge which is identical in format to the payment protocol message. Now the user, while believing that he is executing an age certification protocol, is actually signing a payment message. This 'chosen protocol attack', as it is known, is a very serious threat.

2 Advanced Public Key Protocols

By now you may have concluded that using public key cryptography well is much harder than the conventional, shared-key variety. An analogy I have used is that a jet fighter is more powerful than a spear, but a lot harder to use properly (and a lot easier to kill yourself with). Over the last fifteen years, researchers have thought up an enormous variety of things which one can do with this technology.

We talked about secret sharing and digital cash in the previous lecture. Here are some of the other possible applications:

- electronic elections – these can provide anonymous voting, and often other features too. For example, it might be possible for anyone to verify that all the votes were counted properly, and it might be impossible for any user to prove to a third party which way he voted (so votes cannot reliably be bought);
- playing poker over the telephone. How do two mutually suspicious parties even toss a coin at a distance?
- zero knowledge proof – Alice can prove a theorem to Bob, but Bob can never repeat the proof to Alice;
- identity-based schemes – a user's public key can be made equal to his user name. In this case, a central authority calculates a secret key and issues it to him.

I do not know how many of these I will have time to discuss, but this is a very active field of research and a good introduction can be found in Schneier's *'Applied Cryptography'*. Some of them have been implemented, and some are even becoming politically controversial — for example, tax collectors and law enforcement agencies do not like the idea of untraceable electronic payments!

3 Subliminal Channels

Signature schemes based on discrete log may introduce a new weakness – the subliminal channel. As the message key k is chosen by the signer at random, it is possible for him to encode information in this in various ways that an observer simply cannot detect. For example, he could choose k such that the first t bits of g^k contained a secret message. This would cost 2^t computations, but there are applications in which we would worry about an information leak of even a few bits.

The classic example – which led to the discovery of sibliminal channels – is nuclear arms control. During the Carter administration, the US and the USSR wanted to place sensors in each others' nuclear facilities that would transmit certain information (such as the number of missiles) but not reveal other kinds of information (such as their location: only a minority of silos would hold real missiles). The ability to hide even ten bits of information would enable a missile to be located in a field of 1000 silos and could have enabled the Russians to mount a first strike. The full story of this fascinating incident is told in GJ Simmons, "The History of Subliminal Channels", IEEE J. Selected Areas in Communications v 16 no 4 (May 98) pp 452–462.

4 Summary

Public key algorithms enable us to do all sorts of new and exciting things. However, this flexibility comes at a price. It is even easier to make serious errors when designing public key protocols than it is when working with conventional cryptographic algorithms. Formal verification techniques can be used here as well, but do not provide a complete solution yet because of the large numbers of ways in which the mathematics of these schemes can potentially be exploited.

5 Further Reading

The most comprehensive introduction to public key technology is to be found in Schneier's *Applied Cryptography*, chapters 2–7, 9, 12–14 and 16–17. Simmons' attack on the Tatebayashi-Matsuzaki-Newmann protocol is in *Communications of the ACM* for November 1994, which contains a number of other security articles as well.

6 Problems

1. The RSA cryptosystem has repeatedly been proposed to strengthen a Diffie-Hellman key exchange. Here, Alice encrypts a response sent by Bob to prove that she is present.

   Message 1 $B \rightarrow A :$ N
   Message 2 $A \rightarrow B :$ $\{N\}_{K_A^{-1}}$

   What is wrong with this? How should authentication be used to strengthen a Diffie-Hellman key exchange?

2. How would you repair the TMN and public-key Needham-Schroder protocols?

3. Matsumoto, Kato and Imai proposed a way to speed up secret computations using insecure auxiliary devices. The idea is that a server in a store could help a weak processor, such as a smartcard, to produce an RSA

signature $m^d \pmod n$ without revealing its key d to the server. In the simplest version of the protocol, the smartcard generates integers d_i and binary weights w_i such that

$$d = \sum_{i=0}^{k} d_i w_i \pmod{\phi(n)}$$

It now sends n, the vector d_i and the message m to the server, which calculates $z_i = m^{d_i} \pmod n$ and returns these to the card. The card can now calculate m^d by multiplying together those z_i for which $w_i = 1$. Show that this is insecure (hint: the server sends to the smartcard one or more message messages designed to reveal information about w_i).

4. How would you prevent chosen-protocol attacks?

5. The UK government is implementing the following key establishment protocol for secure email between departments. In order to communicate with Bob, Alice must obtain a 'public receive key' for him from her departmental crypto custodian *TA* and operate on this with a 'secret send key' that *TA* also issues her, along with a certificate on the corresponding 'public send key'. At the other end, Bob will obtain a 'secret receive key' for her from his crypto custodian *TB* and will use this to operate on her 'public send key' whose certificate he will check. It is assumed that the two custodians start off with a shared master key K_{tab}.

   The long term seed key that governs Bob's reception of traffic from all users in the domain of *TA* is $\text{rseed}_{B,A} = \{B\}_{K_{TAB}}$; a secret receive key of the day is derived from this as $\text{SRK}_{B,A,D} = \{D\}_{\text{rseed}_{B,A}}$; and Bob's public receive key of the day for receiving messages from users in the domain of *TA* is $\text{PRK}_{B,A,D} = g_A^{\text{SRK}_{B,A,D}} \pmod{N_A}$. The base g_A and the modulus N_A are fixed by *TA*. Alice's secret send key of the day is similarly derived by encrypting the date by a long term seed key which *TA* gives her: $\text{SSK}_{A,D} = \{D\}_{\text{sseed}_A}$; she generates a shared key of the day with Bob as

   $$k_{A,B,D} = (\text{PRK}_{B,A,D})^{\text{SSK}_{A,D}} \pmod{N_A}$$

   and sends this, along with her public send key of the day $\text{PSK}_{A,D} = g_A^{\text{SSK}_{A,D}} \pmod{N_A}$ and a certificate on this signed by *TA*, to Bob.

   Find as many faults as you can with this protocol.

   It is GCHQ policy that this protocol be used to not just to encrypt content but also to protect signature keys being distributed to public sector workers. What are the possible consequences of this? It is also policy to use this protocol in the National Health Service. What might the effects of that be?