Robustness principles for public key protocols

Ross Anderson and Roger Needham
Cambridge University Computer Laboratory
Pembroke Street, Cambridge, England CB2 3QG

Abstract: We present a number of attacks, some new, on public key protocols. We also advance a number of principles which may help designers avoid many of the pitfalls, and help attackers spot errors which can be exploited.

1 Introduction

Cryptographic protocols are typically used to identify a user to a computer system, to authenticate a transaction, or to set up a key. They typically involve the exchange of about 2–5 messages, and they are very easy to get wrong: bugs have been found in well known protocols years after they were first published. This is quite remarkable; after all, a protocol is a kind of program, and one would expect to get any other program of this size right by staring at it for a while.

A number of remedies have been proposed. One approach is formal mathematical proof, and can range from systematic protocol verification techniques such as the BAN logic [BAN89] to the case-by-case reduction of security claims to the intractability of some problem such as factoring.

Another approach is to try to encapsulate our experience of good and bad practice into rules of thumb; these can help designers avoid many of the pitfalls, and, equally, help attackers find exploitable errors. A recent paper by Abadi and Needham undertook this exercise for cryptographic protocols in general [AN94].

The two approaches — formal proofs and structured design rules — are in many ways complementary. On the one hand, we expect that robust design principles will help us construct protocols which are more amenable to formal verification; on the other hand, protocol errors thrown up by formal analysis may lead us to new insights into the nature of robustness.

Mathematical techniques are also liable to error — quite a few protocols which had been 'proved' secure have been successfully attacked (see, e.g., [PW91] [DB93] [PW95]). These failures are typically due to assumptions which were not made explicit, and we believe that robustness principles are a large part of the solution: if they force us to consider more of the security dependencies, and from a number of aspects, then we will be less likely to produce a 'proof' which neglects a real weakness.

Our goal here is to push the state of the art a little further and deal with more complex protocols, and in particular with public key schemes. Curiously enough, although public key algorithms are based more on mathematics than secret key algorithms, and have much more compact mathematical descriptions, public key protocols have proved much harder to deal with by formal methods.

In this paper, we propose a number of principles. We present a number of new attacks, and give a (hopefully) new perspective on some old ones.

2 The Order of Encryption and Signature

We will start by expanding on the fifth principle from [AN94].

> **Principle 1:** Sign before encrypting. If a signature is affixed to encrypted data, then one cannot assume that the signer has any knowledge of the data. A third party certainly cannot assume that the signature is authentic, so nonrepudiation is lost.

This was motivated by attacks in which the opponent could remove a signature from an encrypted message and replace it with one of his own — X.509 [CCITT88] suffered from such an attack. However, there is an even more powerful attack on several protocols which do encryption before signature, including X.509 and a number of the proposals in ISO CD 11770 [ISO94a].

Suppose that Alice wishes to use RSA [RSA78] with a modulus of 500 - 600 bits to send Bob the message M. The standard technique would be for her to first sign the message with her private key and then encrypt it with his public key. However, suppose that Alice first encrypts M under Bob's public key and then signs it with her private key. Denoting the modulus, public exponent and private exponent of party α by n_α, e_α and d_α, and ignoring hashing (as it makes no difference to our argument), the signed encrypted message would be:

$$\{M^{e_B} \pmod{n_B}\}^{d_A} \pmod{n_A} \tag{1}$$

This is vulnerable, and in a novel way. Since Bob can factor n_B and its factors are only 250–300 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 if he wants to get Alice's 'signature' on a different message, M', he can find x such that

$$[M']^x = M \pmod{n_B} \tag{2}$$

He then registers (xe_B, n_B) as a public key with a certification authority, and claims that the message signed by Alice was not M but M'.

This provides a direct attack on CCITT X.509, in which Alice signs a message of the form $\{T_A, N_A, B, X, \{Y\}^{e_B} \pmod{n_B}\}$ and sends it to Bob. Here T_A is a timestamp, N_A is a serial number, and X and Y are user data. It also breaks the draft ISO CD 11770; there, Y consists of A's name and a random challenge in key agreement mechanism 5, and A's name followed by a session key in key transport mechanisms 2, 5 and 6.

The attack is not limited to RSA: it works with ElGamal too [Elg85], provided this time that Bob can choose his own modulus. Recall that in ElGamal the message m is encrypted to (r, c) where $r = g^k \pmod{p}$, $c = y^k m \pmod{p}$, the message key is k, the recipient's private key is x, and his public key is $y = g^x$. Suppose that Bob selects a so-called 'trapdoor' modulus, under which he can work out discrete logarithms [RLS+92]. Then, for any given m', r and c, he can find a suitable y' such that $(y')^k = m'/c$.

The obvious countermeasure, of requiring all users to share the same modulus, may be politically difficult, as attempts have been made in the past to foist suspect moduli on the user community [And93a].

Key spoofing attacks are also possible on symmetric systems. For example, given a message M and a ciphertext C, the effort required to find a key such that $C = \{M\}_K$ is about 2^{55} when the algorithm used is single key DES, but thanks to the birthday problem it is only 2^{28} or so with double key DES. There are systems where a single, double DES encrypted block is used to authorise a payment, such as described in [And92b] — although that particular system is not vulnerable as one of the two keys is fixed.

With public key systems, key spoofing attacks seem easy to prevent — just always sign before encrypting. However, inverting the order of encryption and signature is a surprisingly common misfeature. A recent internet cash proposal [Oto94] also used it, and Kailar pointed out that this destroyed accountability in the invoicing system [Kai95]. Our attack goes further; it could allow invoices to be forged. It also dents a protocol for anonymous credit cards [LMP94][1].

Encryption before signature can also cause problems for formal verification techniques. The BAN logic ignores the algorithm issues, but at least it will not verify that Alice signed M in equation (1) above as she has no jurisdiction over n_B. Kailar's logic also rejects a signed encrypted message.

However, the verification tools which do try to deal with algorithm properties (such as those discussed in [KMM94]) do not seem able to deal with this attack at all. In order to fix this, the scope of their assumptions may need to be extended; one conventionally worries about the factorisation properties of RSA keys, not their discrete log properties, and the worry about trapdoor primes has been that users might be attacked by authority, rather than by each other.

In any case, it is prudent to sign before encrypting.

3 Spot the Oracle

Nonrepudiation is complicated by the fact that signature and decryption are the same operation in RSA, which many people use as their mental model of public key cryptography. They are actually quite different in their semantics: decryption can be simulated, while signature cannot. By this we mean that an opponent can exhibit a ciphertext and its decryption into a meaningful message, while he cannot exhibit a meaningful message and its signature (unless it is one he has seen previously).

Consider for example the following protocol by Woo and Lam [WL92]:

[1] message 2 in the extension of credit protocol can be arbitrarily manipulated by B_p

Message 1 $A \longrightarrow S$: A, B
Message 2 $S \longrightarrow A$: CB
Message 3 $A \longrightarrow B$: $\{A, N_A\}_{KB}$
Message 4 $B \longrightarrow S$: $A, B, \{N_A\}_{KS}$
Message 5 $S \longrightarrow B$: $CA, \{\{N_A, K_{AB}, A, B\}_{KS^{-1}}\}_{KB}$
Message 6 $B \longrightarrow A$: $\{\{N_A, K_{AB}, A, B\}_{KS^{-1}}\}_{KA}$
Message 7 $A \longrightarrow B$: $\{N_A\}_{K_{AB}}$

Here we are using the standard notation from [BAN89]: CX is a certificate containing the public key KX of participant X; the corresponding private key is KX^{-1}; N_X is a nonce generated by participant X; K_{AB} is a shared secret between Alice and Bob (which it is the purpose of the protocol to generate), and Sam is the key distribution centre.

There are a number of problems with this protocol, including the obvious one that Bob has no assurance of freshness (he does not check a nonce or see a timestamp). However, a subtler and more serious problem is that Alice never signs anything; the only use made of her secret is to decrypt a message sent to her by Bob. The consequence of this is that Bob can only prove Alice's presence to himself — he cannot prove anything to an outsider, as he could easily have simulated the entire protocol run. The effect that such details can have on the beliefs of third parties is one of the interesting (and difficult) features of public key protocols: few of the actual or proposed standards provide a robust nonrepudiation mechanism, and yet there is a substantial risk that many of them may be used as if they did.

We shall return to this topic later. For the meantime let us just say that we must be careful what we mean by 'Bob'. This may be 'whoever controls Bob's signing key', or it may be 'whoever controls Bob's decryption key'. Both keys are written as KB^{-1} in the standard notation, but they are actually rather different.

> **Principle 2:** Be careful how entities are distinguished. If possible avoid using the same key for two different purposes (such as signing and decryption), and be sure to distinguish different runs of the same protocol from each other.

Beaver's attack on Den Boer's oblivious transfer protocol falls into this category: when the same public key primitive is reused in the oblivious transfer context, various sneaky attacks become possible [Bea92]. Also, Landrock recently pointed out that if someone uses the same key in the ISO protocols for signature and zero knowledge proof, there is a massive security failure: the zero knowledge protocol can be used as an oracle to generate signatures [Lan95].

Woo and Lam's protocol also suffers from an oracle problem: if decryption and signature are performed using the same key, then Sam can be impersonated. This is because between messages 4 and 5, he decrypts a nonce N_A sent to him encrypted under his own public key.

Even where keys are only used for one purpose, there may still be an oracle attack; a recent example was found in the documentation for Lotus Notes Internals [Dwo94]. Oracle attacks can be fixed in various ways, such as by explicit typing of nonces, or by using different keys for different purposes (as Lotus apparently do in their current implementations). However, they can sometimes be quite subtle, and an interesting example is the attack found by Simmons on the TMN (Tatebayashi-Matsuzaki-Newmann) scheme.

Here, two users want to do a key exchange, but with a trusted server doing most of the work (we can think of the users as smartcards). If Alice and Bob are the users, and the trusted server Sam can factor N, then the protocol goes as follows:

Message 1 $A \rightarrow S : r_A^3 \pmod N$
Message 2 $B \rightarrow S : r_B^3 \pmod N$
Message 3 $S \rightarrow A : r_A \oplus r_B$

Each party chooses a random number, cubes it, and sends in to Sam. He extracts cube roots, xors the two random numbers together, and sends the result to Alice. The idea is that Alice and Bob can now use r_B as a shared secret key. However, Simmons pointed out that if Charlie and David conspire, or if just one user (David) generates predictable random numbers r_D, then Charlie can get hold of r_B in the following way [Sim94]:

Message 1 $C \rightarrow S : r_B^3 r_C^3 \pmod n$
Message 2 $D \rightarrow S : r_D^3 \pmod n$
Message 3 $S \rightarrow C : r_B r_C \oplus r_D$

We will sum all this up simply as

Principle 3: Be careful when signing or decrypting data that you never let yourself be used as an oracle by your opponent.

4 Count the Bits

We mentioned above the need to distinguish different runs of the same protocol. This means, for example, that systems based on discrete log typically need a fresh message key for each session, which brings us to the topic of subliminal channels.

The message keys in ElGamal type schemes contain various covert channels. For example, 160 of the 320 signature bits in the digital signature standard give security — apparently making the computational security $O(2^{80})$ — but the other 160 bits are available for covert communication [Sim94b].

Counting bits is not always as straightforward, as it may involve specific properties of the public key primitive. One of the earliest examples of an attack on a public key protocol is due to DeMillo and Merritt, who showed that a poker protocol leaked information through quadratic characters [DM83]. A similar attack has been reported by Lomas on a protocol of Mao.

Subliminal channels may seem rather abstract, but counting the precise amount of redundancy can bring us right down into the muddy details of particular implementations. It has been known since the earliest days of public key cryptography [DH76] that digital signatures are inherently vulnerable to attacks by forward search — an attacker applies your public key to a lot of random signatures until (with luck) she gets something which she can pass off as a message from you.

Such attacks can in principle be prevented by putting enough redundancy in each message to be signed. However, it is common to rely on naming information and counters for this purpose, and this can lead to errors — especially if neither the cryptologist nor the system designer pays attention to the other's work.

Consider ISO 11166 [ISO94b]. Here, as in X.509 and ISO CD 11770, encryption is done before signature; but as the RSA exponent is fixed, the key spoofing attack of section 2 above does not work. However, an attacker can just as easily replace the modulus and do a forward search.

How much effort will this take? In ISO 11166, the protected message consists of a key used to authenticate banking transactions, an eight bit key control vector, and a 56 bit counter. However, the standard specifies that if the user receives a count which is higher than the retained value, he should accept it and send a service message confirming the new count. Assuming that the attacker can intercept and discard this message, the counter contains only one bit of real redundancy, and so forging a message is trivial.

The effects of this are surprisingly subtle. For example, public key certificates must be checked anew with every key service message. If they are cached in the local host after checking, then programmers could forge a key service message to their own bank and could then authenticate a bogus transaction.

Another problem with ISO 11166 is that the redundancy in the key certificates is rather low. It is apparently 45 bits for a short certificate, but depends on the redundancy of the namespace for a long certificate; this means that the redundancy will steadily deteriorate. If in future there were 30,000 banks sharing the US banking namespace, then the search effort might be as little as 2^{42} modular multiplications — a large computation, but not large enough to stop a determined attacker. We conclude

Principle 4: Account for all the bits — how many provide equivocation, redundancy, computational complexity, and so on. Make sure that the redundancy you need is based on mechanisms which are robust in the application context, and that any extra bits cannot be used against you in some way.

5 Assume Nothing

We will next look at a number of related types of protocol failure, which are nicely illustrated by a pernicious attack which Burmester found on protocols by Goss, Günter, Yacobi and the EC's RIPE project team [Bur94]. These protocols try to fortify Diffie Hellman key exchange by adding authentication. As an example, we will consider the Goss protocol [Gos90], which is apparently used in the German railway system.

Here, there is a common prime p and a generator g of a high order subgroup of Z_p^*. Each user U has a secret key x_U and a public key $y_U = g^{x_U}$. At each run of the protocol, user U generates the random number r_U. Alice and Bob exchange the message key $y_A^{r_B} \oplus y_B^{r_A}$ as follows:

$$\begin{array}{ll} \text{Message 1 } A \rightarrow B : CA, g^{r_A} & \pmod p \\ \text{Message 2 } B \rightarrow A : CB, g^{r_B} & \pmod p \end{array}$$

Burmester showed that there is a triangle attack: if old session keys can be obtained by an attacker, then Charlie can discover the key shared between Alice and Bob by using suitably chosen values in subsequent exchanges with them. The details can be found in [Bur94].

There is an easier way to look at this attack: Alice supplies g^{r_A} to Bob, and Bob returns to her $(g^{r_A})^{x_B}$ — so Alice can try to send him an arbitrary z and get back z^{x_B}. In fact, she gets back $y_A^{r_B} \oplus z^{x_B}$ and knows only the first of these two terms, but given the key which Bob thought he generated she can work out z^{x_B}. Thus if Bob lets old message keys leak, he will have allowed himself to be used as an oracle for his own secret operation, namely raising numbers to the exponent x_B.

Anyway, Burmester described the flaw in these protocols not as an oracle attack but as a consequence of failing to consider what might happen if a counterparty failed to keep an old message key secret. A number of other protocols fail if a message key is later revealed, and some early examples can be found in [BAN89]; Simmons' attack on the TMN protocol provides another example. We therefore propose as our next principle:

> **Principle 5:** Do not assume the secrecy of anybody else's 'secrets' (except possibly those of a certification authority).

A related error is to make simplifying assumptions about the kind of messages which an opponent might insert in the course of an attack. The weakness in the Goss protocol which we discussed above can also be interpreted in this way: once one sees that the number received from the other party is not necessarily g^r, for some r known to either the other player or an attacker, but can be any number z, then the existence of an oracle attack becomes obvious.

However, there have been other attacks in the same mould which principle 5 does not tackle. Desmedt and Burmester broke a 'proven' secure protocol by showing that the opponent did not have to act in a nice (simulatable) way [DB93], and a number of server assisted signature schemes have also failed in this way [And92b]. We therefore state as a separate principle:

> **Principle 6:** Do not assume that a message you receive has a particular form (such as g^r for known r) unless you can check this.

Next, we have to look at conspiracy attacks on threshold schemes and other multiparty constructions. These have caused a lot of confusion in the past, and perhaps the obvious thing to say would be something like "It is prudent to make explicit the number of conspirators against whom security is claimed".

But we need to go further. One of the present authors proposed a scheme for hiding trapdoors in RSA public keys [And93a] which turned out to be vulnerable to an attack by lattice basis reduction once a certain number of keys had been generated [Kal93]. We have also seen above that some encryption algorithms are vulnerable to key spoofing. So the next principle is a bit more general:

> **Principle 7:** Be explicit about the security parameters of crypto primitives. A key generation routine should be claimed as good for so many keys; a threshold scheme for resistance to so many conspirators; a block cipher for so many blocks; and so on.

Of course, some of the above principles overlap, and Burmester's attack is particularly interesting as it can be construed in different ways — as the consequence of the Goss protocol's failing to observe principles 3, 5 or 6 (at least). However, we are not trying to provide a minimal set of principles; a certain redundancy of robustness concepts is unlikely to do us any harm.

Finally, there are principles which are either too algorithm specific, or too general, for the level of abstraction at which we are trying to operate. Consider for example Coppersmith's attack on NIKS-TAS, a scheme which combines discrete exponentiation with combinatorics [Cop94]. One might formulate a principle that one should shield secrets behind the public key primitive rather than the combinatorics, or one might adopt Coppersmith's own conclusion that in a discrete log scheme one should always have the secret information 'upstairs' and the public information 'downstairs'. However, it is unclear that either of these is general enough for our list.

At higher levels of abstraction, we have 'engineering commonsense' such as the KISS principle ('Keep It Simple Stupid'), which cryptographers often ignore. Many highly complex schemes are proposed for digital cash and other applications, and many of them turn out to be unsound (e.g., [TT94]). Proof is no panacea: several 'proven' secure systems fail because of unexamined assumptions [PW91] [PW95], and others omit to provide desirable properties such as unlinkability [Yac94] and arbitration [Kai95]. Of course, particular schemes may breach one or more of our principles, as they often concatenate a number of public key primitives without hashing or redundancy in order to achieve exotic effects. This is a subject of ongoing research interest.

6 The Explicitness Principle

Looking at the above seven principles, we are led to ask whether there is any overarching principle of which the others are in some sense instances. We propose the following, which one of us put forward in the computer security context in [And94a] and the other in the general protocol context in [AN94].

> **Principle 8:** Robust security is about explicitness; one must be explicit about any properties which can be used to attack a public key primitive, such as multiplicative homomorphism, as well as the usual security properties such as naming, typing, freshness, the starting assumptions and what one is trying to achieve.

With symmetric key algorithms it is often possible to treat the algorithm as a black box, as symmetric block ciphers have a certain amount of muddle in them. For example, it is not common to find that the internals of DES interact malignantly with the way you use it. However, the known asymmetric algorithms are much more structured[2], and depend on fairly straightforward mathematical operations such as integer or matrix multiplication. Thus they are much more likely to interact with other things in the protocols they are used with, and we have to be that much more careful.

Thus it is prudent to hash data before signing it, using a hash function which does not interact with the signature scheme. In an ideal world, signature schemes would be proof against adaptive chosen ciphertext attacks, but in the real world it seems that we can only achieve this by combining hashing with signature, and by being very explicit about what properties of our signature scheme we wish our hash function to mask.

A good example is correlation freedom. Until fairly recently, it was thought sufficient for a hash function to be one-way and collision-free [Dam87]. Then at Crypto 92, Okamoto defined correlation freedom to be the property that we cannot find $M \neq M'$ with $h(M)$ and $h(M')$ agreeing in more bits than we would expect to find from random chance. He conjectured that correlation freedom was strictly stronger than collision freedom, and this was proved in [And93b]. Since then, Vaudenay has shown that MD4 is not correlation free, and Knudsen has established the same for the hasing mode of SAFER K-64 [Knu95].

Here, the explicitness principle is wider than the concept of resistance to adaptive chosen ciphertext attacks. For example, we may also have to protect message keys: in the Schnorr signature scheme [Sch89], we must not have $h(g^r, m)$ equal to $f(k) + h(g^{r+k}, m)$ for any function f which our opponent is able to compute. It is also wider than any possible set of freedom properties (collision freedom, correlation freedom, multiplication freedom, ...) [And93b].

The explicitness principle can be applied to algorithms as well as protocols. We saw above, for example, that algorithm designers should be explicit about the

[2] It was conjectured to one of us by JWS Cassels in 1977 that this was necessarily so, and that asymmetric algorithms would in consequence always be rather fragile.

difficulty of finding key collisions on a given message-ciphertext pair. Another example is the persistence of attacks on hash functions based on modular multiplication, such as that proposed with X.509, where a failure to make the round function sufficiently multiplication free leads to attacks based on techniques such as lattice basis reduction [Cop89]. However extending the explicitness principle too far into the domain of algorithms would take us away from the subject matter of this paper.

Finally, our standard disclaimer: the weaknesses we have discussed do not necessarily imply that any given system based on a protocol criticised above is insecure, as there are many ways to implement compensating controls. However, it is prudent to avoid using standards which are questionable, and which make security depend closely on application detail. Once the application code is brought inside our security perimeter, we lose the advantages of a trusted computing base; we run the risk of unpredictable security failures as documented in [And94a]; and we acquire the legal exposures described in [And94b]. Ignoring prudent design practice can be just as expensive in cryptography as in other branches of engineering.

7 Conclusion

We have tried to extend the prudent engineering principles of Abadi and Needham to the world of public key protocols, which are even more prone than conventional ones to subtle errors, and thus may be even more in need of robustness guidelines. We do not claim that our proposed principles are either necessary or sufficient, just that they are useful; at least, we have found them to be useful both in looking for attacks and in explaining this subject to new students.

Acknowledgement: We are grateful to Martín Abadi for a number of helpful comments.

References

[And92a] RJ Anderson, "Attack on server-assisted authentication protocols", in *Electronics Letters* v **28** no 15 (16th July 1992) p 1473

[And92b] RJ Anderson, "UEPS - A Second Generation Electronic Wallet", *Computer Security — ESORICS 92*, Springer LNCS v **648** in 411–418

[And93a] RJ Anderson, "A practical RSA trapdoor", in *Electronics Letters* v **29** no 11 (27th May 1993) p 995

[And93b] RJ Anderson, "The Classification of Hash Functions", in *Codes and Ciphers* (proceedings of fourth IMA Conference on Cryptography and Coding, December 1993), published by IMA (1995) pp 83–93

[And94a] RJ Anderson, "Why Cryptosystems Fail", in *Communications of the ACM* v **37** no 11 (November 1994) pp 32–40

[And94b] RJ Anderson, "Liability and Computer Security — Nine Principles", in *Computer Security — ESORICS 94*, Springer LNCS v **875** pp 231–245

[AN94] M Abadi, RM Needham, *'Prudent Engineering Practice for Cryptographic Protocols'*, DEC SRC Research Report **125** (June 1 1994)

[Bea92] D Beaver, "How to Break a 'Secure' Oblivious Transfer Protocol", in *Advances in Cryptology — EUROCRYPT '92*, Springer LNCS v **658** pp 284–296

[BAN89] M Burrows, M Abadi, RM Needham, "A Logic of Authentication", in *Proceedings of the Royal Society of London A* v **426** (1989) pp 233–271; earlier version published as DEC SRC Research Report **39**

[Bur94] M Burmester, "On the Risk of Opening Distributed Keys", in *Advances in Cryptology — CRYPTO '94*, Springer LNCS v **839** pp 308–317

[Cop89] D Coppersmith, "Analysis of ISO/CCITT Document X.509 Annex D", submitted to ISO

[Cop94] D Coppersmith, "Attack on the Cryptographic Scheme NIKS-TAS", in *Advances in Cryptology — CRYPTO '94*, Springer LNCS v **839** pp 294–307

[CCITT88] CCITT X.509 and ISO 9594-8, "The Directory — Authentication Framework", CCITT Blue Book, Geneva, March 1988

[Dam87] IB Damgård, "Collision free hash functions and public key signature schemes", in *Advances in Cryptology — EUROCRYPT '87*, Springer LNCS **304** pp 203–216

[Dwo94] C Dwork, "Distributed Computing Column", ACM SIGACT News v 26 mo 1 (Mar 94) pp 17–19

[DB93] Y Desmedt, M Burmester, "Towards Practical 'Proven Secure' Authenticated Key Distribution", in *1st ACM Conference on Computer and Communications Security* (ACM November 1993) pp 228–231

[DH76] W Diffie, ME Hellman, "New Directions in Cryptography", in *IEEE Transactions on Information Theory*, **IT-22** no 6 (November 1976) p 644–654

[DM83] R DeMillo, M Merritt, "Protocols for Data Security", in *IEEE Computer* v **16** no 2 (Feb 1983) pp 39–50

[Elg85] T El-Gamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", in *IEEE Transactions on Information Theory* **IT-31** no 4 (July 1985) pp 469–472

[FS86] A Fiat, A Shamir, "How To Prove Yourself: Practical Solutions to Identification and Signature Problems", in *Advances in Cryptology — CRYPTO 86*, Springer LNCS v **263** pp 186–194

[Gos90] KC Goss, *'cryptographic method and apparatus for public key exchange with] authentication'*, US patent no. 4,956,863 (September 11, 1990)

[ISO94a] ISO DIS 11770, *'Information Technology — Security Techniques — Key Management — Part 3: Mechanisms using asymmetric techniques'*, ISO IST/33/-/2:94/211

[ISO94b] ISO 11166–1:1994, *'Banking — Key management by means of asymmetric algorithms — Part 1: Principles, procedures and formats'*, and *Part 2: Approved algorithms using the RSA cryptosystem'*, 15 November 1994

[Kai95] R Kailar, "Reasoning about Accountability in Protocols for Electronic Commerce", accepted for *Oakland 95*

[Kal93] B Kaliski, "Anderson's RSA trapdoor can be broken", in *Electronics Letters* v **29** no 15 (22nd July 1993) pp 1387–1388

[Knu95] L Knudsen, "A Weakness in SAFER K-64", *this volume*

[KMM94] R Kemmerer, C Meadows, J Millen, "Three Systems for Cryptographic Protocol Verification", in *Journal of Cryptology* v **7** no 2 (Spring 1994) pp 79–130

[Lan95] P Landrock, talk given at Combridge Protocols Workshop, 19–21 April 1995

[LMP94] "Anonymous Credit Cards", SH Low, NF Maxemchuk, S Paul, in *Proceedings of 2nd ACM Conference on Computer and Communications Security* (ACM, Nov 94) pp 108–117

[Oto94] K O'Toole, The Internet Billing Server — Transaction Protocol Alternatives", *Carnegie Mellon University report INI TR 1994-1* (April 26, 1994)

[PW91] B Pfitzmann, M Waidner, "How to Break and repair a 'Provable Secure' Untraceable Payment System", in *Abstracts of Crypto '91* pp 8-14 to 8-19

[PW95] B Pfitzmann, M Waidner, "How to Break Another 'Provably Secure' Payment System", *to appear in proceedings of Eurocrypt 95*

[RLS+92] RA Rueppel, AK Lenstra, ME Smid, KS McCurley, Y Desmedt, A Odlyzko, P Landrock, "The Eurocrypt '92 Controversial Issue — Trapdoor Primes and Moduli", in *Advances in cryptology — EUROCRYPT '92*, Springer LNCS v **658** pp 194–199

[RSA78] RL Rivest, A Shamir, L Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", in *Communications of the ACM* **21** (1978) pp 120–126

[Sch89] CP Schnorr, "Efficient identification and signatures for smart cards", in *Advances in Cryptology — CRYPTO '89*, Springer LNCS **435**, pp 239–251

[Sim94a] GJ Simmons, "Cryptanalysis and Protocol Failures", in *Communications of the ACM* v **37** no 11 (November 1994) pp 56–65

[Sim94b] GJ Simmons, "Subliminal Channels; Past and Present", in *European Transactions on Telecommunications* v **5** no 4 (July/Aug 1994) pp 459–473

[TMN89] M Tatebayashi, N Matsuzaki, DB Newman, "Key distribution protocol for digital mobile communication systems", in *Advance in Cryptology — CRYPTO '89*, Springer LNCS **435** pp 324–333

[TT94] L Tang, D Tygar, "A fast off-line electronic currency protocol for smart cards", in *proceedings of the First Smart Card Research and Advanced Application Conference* (University of Lille, Oct 94) pp 89–100

[Vau94] S Vaudenay, "On the need of multipermutations - Cryptanalysis of MD4 and SAFER", in *'Fast Software Encryption'*, proceedings of KU Leuven workshop on cryptographic algorithms (Springer, to appear)

[WL92] TYC Woo, SS Lam, "Authentication for Distributed Systems", in *IEEE Computer* (January 1992) pp 39–52

[Yac94] Y Yacobi, "Efficient Electronic Money", in *Preproceedings of Asiacrypt 94* pp 131–140

This article was processed using the LaTeX macro package with LLNCS style