# A Survey of Authentication Protocol Literature: Version 1.0

John Clark and Jeremy Jacob

17 November 1997

# Contents

# 1 Introduction

## 1.1 Background

The past two decades have seen an enormous increase in the development and use of networked and distributed systems, providing increased functionality to the user and more efficient use of resources. To obtain the benefits of such systems parties will cooperate by exchanging messages over networks. The parties may be users, hosts or processes; they are generally referred to as *principals* in authentication literature.

Principals use the messages received, together with certain modelling assumptions about the behaviour of other principals to make decisions on how to act. These decisions depend crucially on what validity can be assumed of messages that they receive. Loosely speaking, when we receive a message we want to be sure that it has been created recently and in good faith for a particular purpose by the principal who claims to have sent it. We must be able to detect when a message has been created or modified by a malicious principal or intruder with access to the network or when a message was issued some time ago (or for a different purpose) and is currently being replayed on the network.

An authentication protocol is a sequence of message exchanges between principals that either distributes secrets to some of those principals or allows the use of some secret to be recognised [26]. At the end of the protocol the principals involved may deduce certain properties about the system; for example, that only certain principals have access to particular secret information (typically cryptographic keys) or that a particular principal is operational. They may then use this information to verify claims about subsequent communication, for example, a received message encrypted with a newly distributed key must have been created after distribution of that key and so is *timely*.

A considerable number of authentication protocols have been specified and implemented. The area is, however, remarkably subtle and many protocols have been shown to be flawed a long time after they were published. The Needham Schroeder Conventional Key Protocol was published in 1978 [87] and became the basis for many similar protocols in later years. In 1981, Denning and Sacco demonstrated that the protocol was flawed and proposed an alternative protocol [42]. This set the general trend for the field. The authors of both papers suggested other protocols based on public key cryptography (see section 2). In 1994 Martin Abadi demonstrated that the public key protocol of Denning and Sacco was flawed [1]. In 1995, Lowe demonstrated an attack on the public key

protocol of Needham and
Schroeder (seventeen years after its publication). In the intervening years a whole host of protocols have been specified and found to be flawed (as demonstrated in this report).

This report describes what sorts of protocols have been specified and outlines what methods have been used to analyse them. In addition, it provides a summary of the ways in which protocols have been found to fail. There is a large amount of material in the field and the main body of this document is intended as a concise introduction to and survey of the field. Some types of protocol are given little detailed attention, particularly those which rely on number-theoretic properties for their security. It is envisaged that future editions of this report will provide a complete coverage. An annotated bibliography is included to guide the reader. Since authentication relies heavily on encryption and decryption to achieve its goals we also provide a brief review of elements of cryptography.

## 1.2 A Protocols Resource

Authentication never stands still! This report is intended as a compendium of useful information related to authentication. Hopefully, this will be useful to researchers and protocol designers alike. However, the authors hope to make this a "living document" and update it as comments from the community are received. The authors [1] welcome suggestions for inclusions in future editions (omissions, necessary corrections, new protocols, new attacks etc.)

[1] jac@cs.york.ac.uk, jeremy@cs.york.ac.uk

# 2 Cryptographic Prerequisites

## 2.1 General Principles

Cryptographic mechanisms are fundamental to authentication protocols. Suppose that we have some message text $P$ which we wish to transmit over the network. $P$ is generally referred to as *plaintext* or a *datagram*. A cryptographic algorithm converts $P$ to a form that is unintelligible to anyone monitoring the network. This conversion process is called *encryption*. The unintelligible form is known as *ciphertext* or a *cryptogram*. The precise form of the cryptogram $C$ corresponding to a plaintext $P$ depends on an additional parameter $K$ known as the *key*.

The intended receiver of a cryptogram $C$ may wish to recover the original plaintext $P$. To do this, a second key $K^{-1}$ is used to reverse the process. This reverse process is known as *decryption*. Encryption and decryption are depicted in figure 1.

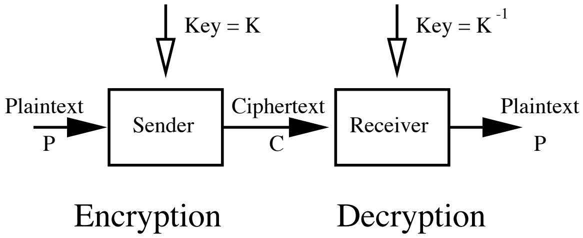


Figure 1: Encryption and Decryption

The classes of encryption and decryption algorithms used are generally assumed to be public knowledge. By restricting appropriately who has access to the various keys involved we can limit the ability to form ciphertexts and the ability to determine the plaintexts corresponding to ciphertexts.

## 2.2 Symmetric Key Cryptography

In **symmetric key cryptography** the encryption key $K$ and the decryption key $K^{-1}$ are easily obtainable from each other by public techniques. Usually they are identical and we shall generally assume that this is the case. The key $K$ is used by a pair of principals to encrypt and decrypt messages to and from each other. Of course, anyone who holds the key can create ciphertexts corresponding to arbitrary plaintexts and read the contents of arbitrary ciphertext messages. To ensure security of communication this key is kept secret between the communicating principals. Following established convention we shall use the notation *Kab* to denote a key intended for communication between principals *A* and *B* using a symmetric key cryptosystem.

### 2.2.1 Classical Cryptography

Classical cryptography has used symmetric keys. Typically classical ciphers have been either *substitution* or *transposition* ciphers (or a mixture) and have worked on text characters. A substitution cipher substitutes a ciphertext character for a plaintext character. A transposition cipher shuffles plaintext characters. The precise substitutions and transpositions made are defined by the key. Examples include simple, homophonic, polyalphabetic and polygram substitution ciphers and simple permutation ciphers (e.g. where successive groups of N characters are permuted in the same way). Elements of transposition and substitution are included in modern day algorithms too. It is not our intention to survey classical approaches to cryptography. They are well documented already [41, 99]. An elementary introduction has been produced by Willet [114].

### 2.2.2 Modernday Cryptography

Modernday symmetric key algorithms are principally *block ciphers* or *stream ciphers*.

A block cipher will encrypt a block of (typically 64 or 128) plaintext bits at a time. The best known block cipher is the ubiquitous Data Encryption Standard [45], universally referred to as DES. This has been a hugely controversial algorithm. The controversy has centred on whether the effective key length (56 bits – reduced from 128 at the insistence of the National Security Agency) is really sufficient to withstand attacks from modern-day computing power (see Wiener [113] for details), and over the design of elements called *S-boxes* (the design criteria were not made public). The reader is referred to [101] for details. It is worth noting that the algorithm is remarkably resistant to attack using the *published* state-of-the-art cryptanalysis technique known as differential cryptanalysis discovered by Biham and Shamir in 1988. As revealed by Coppersmith in 1994 [38] this was because the technique was known to the designers of DES back in 1974! Of course, in this survey we can only comment on what is publicly known.

Other examples of block ciphers are MADRYGA (efficient for software implementation and operates on 8-bit blocks), NEWDES (operates on 64-bit blocks but with a 120-bit key), FEAL-N, RC2 and RC4 (by Ronald Rivest) and IDEA (by Lai and Massey). Schneier has written a readable account of the IDEA algorithm [98]. A very good overview of block ciphers (and others) can be found in Schneier's general cryptography text [99].

### 2.2.3 Modes of Block Cipher Usage

There are several modes in which a block cipher can be used. Principal ones are:

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Cipher Feedback Mode (CFB)
- Output Feedback Mode (OFB)

ECB is the simplest mode. Consecutive blocks of plaintext are simply encrypted using the algorithm. Thus, identical blocks of plaintext are always encrypted in the same way (with the same result). Its security needs to be questioned for specific contexts. An analyst may be able to build up a codebook of plaintext-ciphertext pairs (either known or because he can apply cryptanalytic methods to derive the plaintexts). Also, it is possible to modify messages (e.g. by simply replacing an encrypted block with another).

Cipher Block Chaining (CBC) is a relatively good method of encrypting several blocks of data with an algorithm for encrypting a single block. It is one mode in which the widely used Data Encryption Standard (DES) can be employed. Block $i$ of plain text is exclusively-ored (hereafter XORed) with block $i-1$ of ciphertext and is then encrypted with the keyed block encryption function to form block $i$ of ciphertext.

For example, with initialisation block $I$ the encryption of message block sequence $P_1P_2\dots P_n$ with key $K$ denoted by $E(K : P_1P_2\dots P_n)$ is given by

$$E(K : P_1P_2\dots P_n) = C_0C_1C_2\dots C_n$$

where

$$C_0 = I$$
$$\forall i, i > 0 \bullet C_i = e(K : (C_{i-1} \oplus P_i))$$

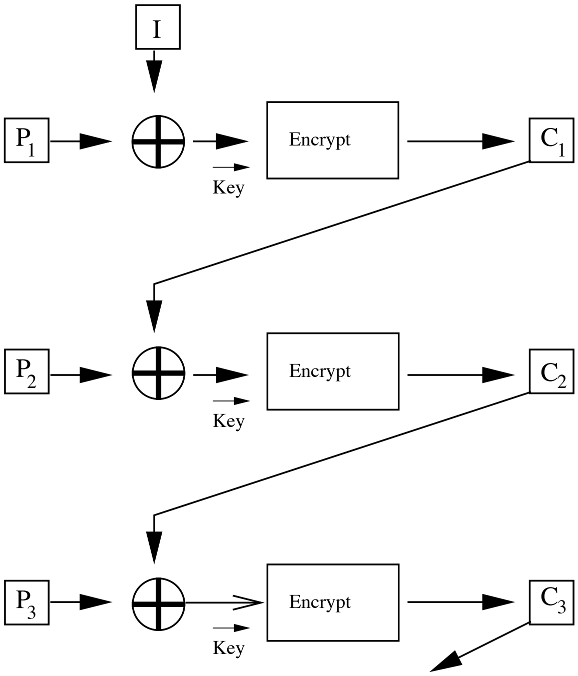


Figure 2: Cipher Block Chaining

Here, $e(K:)$ is the block encryption function used with key $K$. The encryption process is shown in figure 2.

Successive ciphertext blocks are decrypted using the keyed block function $d(K:)$ according to the rule

$$\forall i, i > 0 \bullet P_i = C_{i-1} \oplus d(K : C_i)$$

Thus, for any successive pair of ciphertext blocks we can recover the plaintext block corresponding to the second (provided we have the key).

If we choose a different initial block $I$ in each case then even identical plaintext messages will have different ciphertexts. It is widely acknowledged that non-repeating initial blocks are essential for adequate preservation of confidentiality (unless the first block in a message is always different in which case it is known as a *confounding block*). Authors differ as to whether they should be passed between communicating parties in the clear (which Schneier [99] thinks is fine) or encrypted ( as recommended by Davies and Price [39]). Voydock and Kent [112] address many aspects of initial block usage insisting that they should be pseudo-random for CBC. The rationale given there and in various other texts is incomplete or simply wrong. For example Schneier states that an initial block can be a serial number that increments after each message but does not repeat. Clark and Jacob [36] have shown that such an approach is potentially disastrous; they show how for the most celebrated authentication protocol of all, adoption of this approach would allow a third party to create the ciphertext for an *arbitrary* message without having access to the key!

In certain network applications it is useful to be able to transmit, receive and process data chunks of size less than the block size (e.g. the processing of character-sized chunks from a terminal). In such cases Cipher Feedback mode (CFB) might be used. Figure 3 is based on a figure by Schneier [99] and shows an 8-bit CFB with a 64-bit block algorithm. Here the contents of a shift register are initialised with some value. The contents of the shift register are encrypted as a block, and the leftmost byte of the result is XORed with the next plaintext byte to produce a ciphertext byte. The contents of the register are now shifted left by 8 bits and the most recently created ciphertext byte is placed in the rightmost byte of the register and the procedure repeats. The decryption procedure is easily obtained.

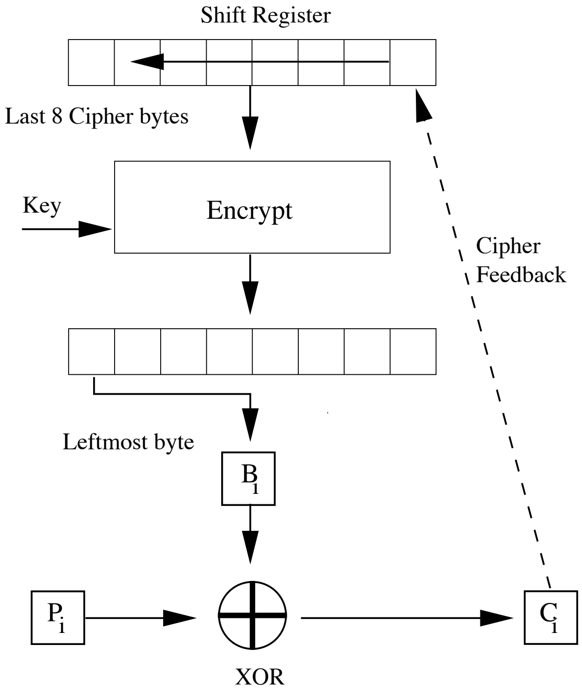


Figure 3: Cipher Feedback Mode

Output Feedback mode (OFB) is shown similarly in figure 4. Here, it is the leftmost byte of the direct output of the encryption function that is fed back into the shift register (other sizes are possible).

Schneier states that the initialisation vectors for CFB and OFB should be different for each message encrypted, though there is no additional benefit from sending them encrypted [99]. Voydock and Kent disagree [112].

The error propagation properties of the different modes of encryption vary but are not detailed here. The reader is referred to Schneier [99] or Davies and Price [39] for details.

Other modes are possible, e.g. Counter mode (like OFB but with the contents of the register simply incremented each time, i.e. no feedback), Block Chaining mode (where the input to the encryption is the XOR of all previous ciphertext blocks and the current plaintext block) and Propagating Cipher Block Chaining (where the input to the encryption is the XOR of the current and the immediately previous plaintext blocks and all previous ciphertext blocks). There are a variety of other modes which are somewhat esoteric; we shall not describe them here.

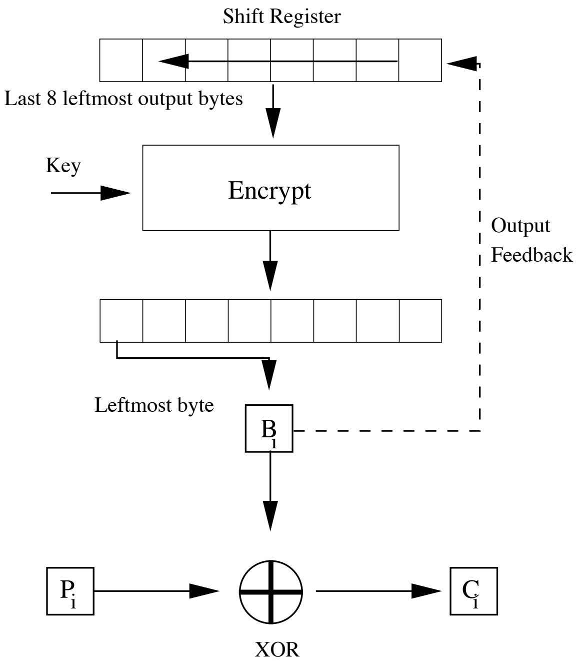


Figure 4: Output Feedback Mode

### 2.2.4 Stream Ciphers

Stream ciphers encrypt one bit of plaintext at a time. The usual approach is to generate a bit stream and to XOR successive bits with successive bits of plaintext. Clearly we should wish the bit-stream produced to be as random as possible. Indeed, a vast amount of work into pseudo-random stream generation has been carried out (see [99]). The streams produced depend on a key in some way (if identical streams were produced each time then cryptanalysis becomes easy). A keystream generator comprises a finite state machine and an output function. Figure 5 shows two basic approaches to bit-stream generation: output feedback mode (where the value of the key affects the next state) and the output function is pretty straightforward; and Counter mode (where the key affects the output function and the next state is straightforward, typically a counter increment).

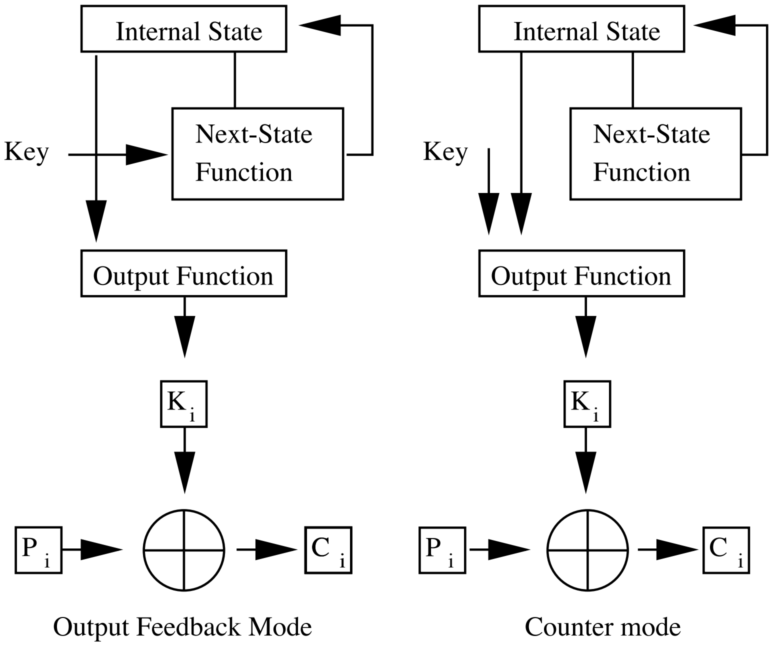


Figure 5: Stream Cipher Approaches

It is also possible to use block ciphers as keystream generators (e.g. use Counter Mode and select the leftmost bit of the encrypted block output). For details of the above see Schneier [99].

## 2.3 Public Key Cryptography

In public key cryptography there is no shared secret between communicating parties. The first publication on the topic was the classic paper by Whitfield Diffie and Martin Hellman in 1976 [44]. In *public key encryption* each principal $A$ is associated with some key pair $(Ka, Ka^{-1})$. The *public key* $Ka$ is made publicly available but the principal $A$ does not reveal the *private key* $Ka^{-1}$. Any principal can encrypt a message $M$ using $Ka$ and only principal $A$ can then decrypt it using $Ka^{-1}$. Thus, the secrecy of messages to $A$ can be ensured.

Some public key algorithms allow the private key to be used to encrypt plaintext with the public key being used to decrypt the corresponding ciphertext. If a ciphertext $C$ decrypts (using $Ka$) to a meaningful plaintext message $P$ then it is assumed that the ciphertext must have been created by $A$ using the key $Ka^{-1}$. This can be used to guarantee the *authenticity* of the message. The most widely known public key algorithm that allows such use was developed by Rivest, Shamir and Adleman [92] and is universally referred to as RSA. Such algorithms are often said to provide a *digital signature* capability. Simply *encrypting* using a private key does not constitute a signature. Various checks must also be made by the receiver (see Gollman [49]).

The RSA algorithm [92] works as follows:

1. pick two large primes $p$ and $q$, let $n = p * q$

2. choose $e$ relatively prime to $\phi(n) = (p-1)(q-1)$

3. use Euclid's algorithm to generate a $d$ such that $e * d = 1 \bmod \phi(n)$

4. make the pair $(n, e)$ publicly available – this is the public key. The private key is $d$.

5. a message block $M$ is now encrypted by calculating $C = M^e \bmod n$.

6. the encrypted block $C$ is decrypted by calculating $M = C^d \bmod n$.

Here encryption and decryption are the same operation (modular exponentiation).

A sender $A$ can communicate with $B$ preserving secrecy and ensuring authenticity by first signing a message using his own private key $Ka^{-1}$ and then encrypting the result using $B$'s public key $Kb$. $B$ uses his private key to decrypt and then uses $A$'s public key to obtain the original message.

Public key algorithms tend to rely on the (supposed) computational difficulty of solving certain problems (e.g. finding discrete logarithms for the Diffie Hellman algorithm and finding prime factors for RSA). Again, key length is an issue. Computing power is increasing rapidly and there have been significant advances. For example, ten years ago 512 bit keys for

RSA were thought to be very secure; today 512 bits is considered a minimum requirement (and 1024 bits is often recommended). Sheer processing capability also affects the usability of public key encryption. Public key algorithms are generally much slower than symmetric key algorithms. Schneier [99] gives a good account of relative speeds of algorithms.

There are some very useful and informative papers that deal (at least in part) with public key cryptography. Hellman provides an excellent introduction to public key cryptography and the underlying mathematics [58]. Willet provides a much higher level view [115]. Gordon [56] provides a good but simple introduction. Diffie provides an exciting account of the first decade of public key cryptography [43] with a particularly good account of the attacks on knapsacks. Brickell and Odlyzko provide an account of various attacks on public key systems (and others) [25]. Other aspects are covered in Massey's informative general paper on cryptology [81].

## 2.4 One-way Hash Algorithms

We shall often require evidence that a message that has been sent has not been subject to modification in any way. Typically this is carried out using a hash function. A hash function H when applied to a message M yields a value H(M) of specific length known as the hash value of that message. H(M) is often referred to as a *message digest*. The mapping of messages to digests is one-way; given M and H(M) it should be computationally infeasible to find M′ such that H(M′)=H(M). The digest is a form of reduced message calculated using a publicly known technique. A receiver of a message can check whether a message and a corresponding digest agree. Hash functions are largely intended for use in conjunction with cryptography to provide signatures.

If $M$ is a message then $A$ can provide evidence to $B$ that he created it, and that it has not been tampered with, by calculating $E(Kab : H(M))$ and sending the message M together with the newly calculated encrypted hash value. On receiving the message, $B$ can calculate $H(M)$ and then $E(Kab : H(M))$ and check whether the value agrees with the encrypted hash value received. Since the amount of encryption is small this is a quite efficient means to demonstrate authenticity (assuming $A$ and $B$ do not fake messages from the other).

## 2.5 Notational Conventions

In this report we shall use the notation $E(K : M)$ to denote the result of encrypting message plaintext $M$ with key $K$.

A protocol run consists of a sequence of messages between principals and will be described using the standard notation. Principals are generally denoted by capitals such as $A$, $B$ and $S$ (for a server). The sequence of messages

$$\begin{array}{lcl} (1)\ A \rightarrow B & : & M_1 \\ (2)\ B \rightarrow S & : & M_2 \\ (3)\ S \rightarrow B & : & M_3 \end{array}$$

denotes a protocol in which $A$ sends $M_1$ to $B$, $B$ then sends $M_2$ to $S$ who then sends $M_3$ to $B$. Attacks on protocols often involve some mischievous principal pretending to be another. We denote a mischievous principal by $Z$. The notation $Z(A)$ denotes the principal $Z$ acting in the role of $A$. $Z$ has unfettered access to the network medium and may place at will messages onto the net claiming to be sent from $A$ and intercepting messages destined for $A$ (and possibly removing them).

A number generated by a principal $A$ is denoted by $Na$. Such numbers are intended to be used *only once* for the purposes of the current run of the protocol and are generally termed **nonces**. We shall sometimes refine the notion of a nonce to include a *timestamp* and distinguish between sequence numbers or genuinely pseudo-random nonces. Such distinctions are made in, for example, the ISO entity authentication standards (see [61]).

A message may have several components; some will be plaintext and some will be encrypted. Message components will be separated by commas. Thus

$$(1)\ A \rightarrow B \quad : \quad A, E(Kab : Na)$$

denotes that in the first message of the protocol $A$ sends to $B$ the message whose components are a principal identifier $A$ together with an encrypted nonce $E(Kab : Na)$.

# 3 Protocol Types

In this section we provide an overview of various forms of authentication protocol in use today. At the highest level we have categorised them according to the principal cryptographic approach taken, i.e. symmetric key or public key. We distinguish also between those that use (one or more) trusted third parties to carry out some agreed function and those that operate purely between two communicating principals that wish to achieve some mode of authentication. There are further distinctions that can be made: the number of messages involved in the protocols (e.g. one-pass, two-pass, three-pass etc.) and whether one principal wishes to convince the second of some matter (one-way or unilateral authentication) or whether both parties wish to convince each other of something (two-way or mutual authentication). These distinctions are also made by the ISO entity authentication standards (see [61]).

## 3.1 Symmetric Key Without Trusted Third Party

Perhaps the simplest (and yet effective) example in this class is the ISO One-pass Symmetric Key Unilateral Authentication Protocol [62] (see also 6.1.1) shown below. It consists of the single message:

(1) $A \rightarrow B \quad : \quad Text2, E(Kab : [Ta|Na], B, Text1)$

Here the text fields shown are optional; their use is implementation specific (and we shall ignore them in this discussion). We can see that the *claimant* $A$ (i.e. the one who wishes to prove something) sends an encrypted message containing a nonce and the identifier of the *verifier* (i.e. the principal to whom the claim is made). The nonce may be a timestamp $Ta$ or a sequence number $Na$ depending on the capabilities of the environment and the communicating principals. On receiving this message, $B$, who believes that the key $Kab$ is known only to himself and $A$, may deduce that $A$ has recently sent this message if the sequence number is appropriate or if the timestamp has a recent value. Note here that if a malicious principal has unfettered access to the network medium then use of sequence numbers will be insufficient (since he can record message (1), prevent $B$ from receiving it, and replay it to $B$ at a later time).

The best-known protocols that do not use a trusted third party are simple challenge-response mechanisms. One principal $A$ issues data to a second principal $B$. $B$ then carries out some transformation and sends the result to $A$ who checks to see if the appropriate transformation has occurred. Figure 6 shows a simple challenge-response protocol. In this case the nonce *Na* should be random. If the nonce were a sequence number, or were otherwise predictable, a malicious principal could issue the next nonce value to *B* and record the response. When *A* genuinely issued the same nonce value at a later date the intruder could replay *B*'*s* earlier response to complete the protocol. *A* could conclude only that the message he receives was created at *some* time by *B* (but not necessarily in response to his most recent challenge).

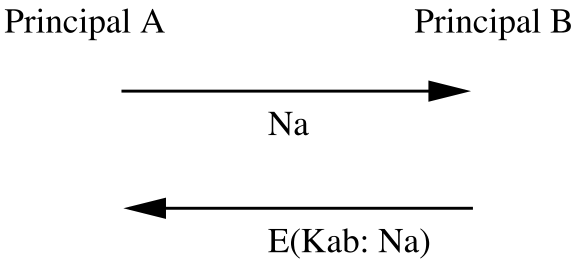


Figure 6: A Challenge Response Protocol

There are other variations on the challenge-response theme. Sometimes the challenge is encrypted, sometimes not; sometimes it is random, sometimes predictable (but never before used). Gong highlights many issues associated with the use of nonces for such purposes [53].

The ISO Two-Pass Unilateral Authentication Protocol is described later in this document (see 6.1.2). The ISO Two- and Three-Pass Mutual Authentication Protocols are described in sections 6.1.3 and 6.1.4 respectively.

Another approach to ensuring authenticity uses cryptographic check functions. Essentially, a message is sent together with some summary or digest calculated using a hash function using a shared key. Examples are given in section 6.2. Examples can be found in Part 4 of the ISO entity authentication standard [64].

## 3.2 Symmetric Key With Trusted Third Party

Symmetric key protocols that use a trusted third party (TTP) are by far the most numerous in the literature. The most celebrated protocol of all time, the Needham Schroeder Symmetric Key Authentication protocol [87] is described below:

$$
\begin{array}{lll}
(1)\ A \rightarrow S & : & A, B, Na \\
(2)\ S \rightarrow A & : & E(Kas : Na, B, Kab, E(Kbs : Kab, A)) \\
(3)\ A \rightarrow B & : & E(Kbs : Kab, A) \\
(4)\ B \rightarrow A & : & E(Kab : Nb) \\
(5)\ A \rightarrow B & : & E(Kab : Nb - 1)
\end{array}
$$

In this protocol $A$ requests from the server $S$ a key to communicate with $B$. He includes a random nonce $Na$ generated specially for this run of the protocol. This nonce will be used by $A$ to ensure that message (2) is timely. $S$ creates a key $Kab$ and creates message (2). Only $A$ can decrypt this message successfully since he possesses the key $Kas$. In doing so he will obtain the key $Kab$ and check that the message contains the nonce $Na$. $A$ passes on to $B$ the encrypted message component $E(Kbs : Kab, A)$ as message (3).

Principal $B$ decrypts this message to discover the key $Kab$ and that it is to be used for communication with $A$. He then generates a nonce $Nb$, encrypts it (using the newly obtained key), and sends the result to $A$ as message (4).

Principal $A$, who possesses the appropriate key $Kab$ , decrypts it, forms $Nb - 1$, encrypts it and sends the result back to $B$ as message (5). $B$ decrypts this and checks the result is correct. The purpose of this exchange is to convince $B$ that $A$ is genuinely operational (and that message 3 was not simply the replay of an old message).

At the end of a correct run of the protocol, both principals should be in possession of the secret key $Kab$ newly generated by the server $S$ and should believe that the other principal has the key. Rather, this is what the protocol is intended to achieve. We shall show in section 4.1 that it is in fact flawed.

There have been many other protocols that have used a trusted third party to generate and distribute keys in a similar way: the Amended Needham-Schroeder Protocol [88] (see 6.3.4), the Yahalom Protocol (see 6.3.6), the Otway-Rees Protocol [91] (see also 6.3.3) which is essentially the same as the Amended Needham-Schroeder Protocol. Woo and Lam provide several authentication protocols [116, 117] (6.3.10). Other examples include those by Gong and Carlsen's secret key initiator protocols (for mobile phone networks) (6.10.2 and 6.3.7) and the ISO Four- and Five Pass Mutual Authentication Protocols [62] (6.3.8 and 6.3.9).

Denning and Sacco suggested fixing problems in the Needham Schroeder protocol using timestamps. The Denning Sacco Conventional

Key Protocol replaces the first three messages of the Needham Schroeder protocol with:

(1) $A \rightarrow S \ : \ A, B$
(2) $S \rightarrow A \ : \ E(Kas : B, Kab, T, E(Kbs : A, Kab, T))$
(3) $A \rightarrow B \ : \ E(Kbs : A, Kab, T)$

Here, $T$ is a timestamp generated by $S$. $A$ and $B$ can check for timeliness of messages (2) and (3) (i.e. the timestamp must be within some window centred on the respective local clock time).

Third parties may be trusted for activities other than key generation and distribution. Consider the Wide Mouthed Frog Protocol due to Burrows (but not for use in real systems) [26]:

(1) $A \rightarrow S \ : \ A, E(Kas : Ta, B, Kab)$
(2) $S \rightarrow B \ : \ E(Kbs : Ts, A, Kab)$

$A$ is trusted to generate a session key $Kab$. On receiving message (1) $S$ checks whether the timestamp $Ta$ is "timely" and, if so, forwards the key to $B$ with its own timestamp $Ts$. $B$ checks whether the message (2) has a timestamp that is later than any other message it has received from $S$. Here the server $S$ effectively performs a *key translation* service (providing also trusted timestamping). Davis and Swick provide more key translation service facilities [40].

Some protocols allow keys to be reused in more than one session. These are typically two-part protocols. The first part involves a principal $A$ obtaining a 'ticket' for communication with a second principal $B$. The ticket generally contains a session key and is encrypted so that only the receiver $B$ can decrypt it. In the second part of the protocol $A$ presents the ticket to $B$ when he wishes to communicate; he may do this on several occasions (until the ticket expires). These are usually called *repeated authentication protocols*. Such protocols have been devised by Kehne *et al* [68] and also by Neuman and Stubblebine [90].

## 3.3 Public Key

Protocols using public key cryptography find numerous applications in authentication but the speed of encryption and decryption using public key algorithms has prevented their widespread use for general communication; for example, Schneier states that RSA encryption is about 100 times slower than DES when both are implemented in software (the fastest hardware implementation of RSA has a throughput of 64 Kbaud). However, exchanging symmetric encryption keys using public key cryptography provides an excellent use of the technology and several such distribution schemes have been created.

Needham and Schroeder proposed the following protocol in their classic work [87]:

(1) $A \to S \ : \ A, B$
(2) $S \to A \ : \ E(Ks^{-1} : Kb, B)$
(3) $A \to B \ : \ E(Kb : Na, A)$
(4) $B \to S \ : \ B, A$
(5) $S \to B \ : \ E(Ks^{-1} : Ka, A)$
(6) $B \to A \ : \ E(Ka : Na, Nb)$
(7) $A \to B \ : \ E(Kb : Nb)$

Here, we see how use is made of a trusted server $S$, generally called a *certification authority*, that stores the public keys of the various principals and distributes them on request sealed under its own private key $Ks^{-1}$. The certification authority's public key is generally assumed known to the principals. Messages (1), (2) and (5), (6) are used by $A$ and $B$ to obtain each other's public keys. Message (3) is encrypted under $B's$ public key and so can only be decrypted successfully by $B$. It contains a challenge $Na$ together with $A's$ identifier. $B$ decrypts this to obtain the challenge, forms a challenge of his own $Nb$ and encrypts both challenges under $A's$ public key and sends the result as message (6). $A$ then decrypts message (6). Since only $B$ could have obtained the information necessary to send this message $A$ knows that $B$ is operational and has just responded to his recent challenge. $A$ then encrypts $B's$ challenge $Nb$ using B's public key $Kb$ and sends message (7). $B$ then decrypts and checks that it contains his challenge and concludes that $A$ is operational and indeed initiated the protocol. This protocol (and the reasoning given above) has only recently been shown to be flawed [74].

Some key distribution protocols use public key cryptography, for example Digital's SPX (see Schneier's book [99] or Woo and Lam [117]). The draft CCITT X.509 standard [29] uses public key cryptography for authenticated communication. The ISO authentication framework makes extensive use of public key cryptography.

Denning and Sacco provide an example of how to use public key cryptography to distribute session keys [42]. Martin Abadi noticed in 1994 that it was terribly flawed [1].

Public key cryptography may also be used to provide digital signatures. RSA [92] can be use to sign a message by encrypting under the private key. It can also be used to sign a hash value of a complete message. The actual message can also be sent in the clear with the encrypted hash value appended. A major alternative to the use of RSA developed, amid some controversy, by the United States National Security Agency (NSA) is the Digital Signature Algorithm. It is based on El Gamal encryption. Schneier provides a good account of the algorithm [99] and a good journalistic account of the controversy can be found in the paper by Adam [2]. Other digital signatures schemes include ESIGN, McEliece (based on algebraic coding theory). Akl provides a good tutorial guide to digital signatures in general [3].

## 3.4 Hybrid Protocols

There are some protocols that use both public and symmetric key cryptography. An example of such is the unusual (but seemingly very effective) Encrypted Key Exchange (EKE) protocol by Bellovin and Merritt [15]. This protocol is unusual in that it uses symmetric key cryptography to distribute 'public' keys. It also seems to tolerate fairly poor mechanisms of symmetric encryption.

## 3.5 Other Forms of Protocol

There are many other types of authentication protocol. For example, protocols that deal with non-repudiation, secret voting, anonymous transactions, anonymous signatures etc. The reader is referred to Schneier for details [99]. Examples of various international standard protocols can be found in [61], [62], [63], [64], [65]. Recent protocols include a beacon based protocol by Seberry *et al* [66] and a robust password exchange protocol by Hauser *et al* [57]. Liebl [73] provides an overview of authentication protocols (in less detail than here).

## 3.6 General

There are many applications of authentication technology that are not discussed above. Simmons provides an example of the need for authenticity in the face of a very hostile enemy for the purposes of verifying nuclear test ban treaties [100]. Anderson provides an indication of how electronic payment systems work [9]. The same author discusses societal and legal aspects of cryptographic technology [8], [7].

# 4 Attacking Authentication protocols

In this section we detail various ways in which protocols fail and give examples.

## 4.1 Freshness Attacks

A freshness attack occurs when a message (or message component) from a previous run of a protocol is recorded by an intruder and replayed as a message component in the current run of the protocol. The classic example of such an attack occurs in the Needham Schroeder conventional (symmetric) key protocol described in section 3.2.

At the end of a correct run of the protocol, each principal should be in possession of the secret key $Kab$ newly generated by the server $S$ and believe that the other has the key. That is what the protocol is *intended* to achieve. In 1981, Denning and Sacco demonstrated that the protocol was flawed [42]. Consider message (3). Although $B$ decrypts this message and (if it is indeed well-formed) assumes legitimately that it was created by the server $S$, there is nothing in the message to indicate that it was actually created by *S as part of the current protocol run*. Thus, suppose a previously distributed key $K'ab$ has been compromised (for example, by cryptanalysis) and is known to an intruder $Z$. $Z$ might have monitored the network when the corresponding protocol run was executed and recorded message (3) consisting of $E(Kbs : K'ab, A)$. He can now fool $B$ into accepting the key as new by the following protocol (omitting the first two messages):

(3) $Z(A) \rightarrow B \quad : \quad E(Kbs : K'ab, A)$
(4) $B \rightarrow Z(A) \quad : \quad E(K'ab : Nb)$
(5) $Z(A) \rightarrow B \quad : \quad E(K'ab : Nb - 1)$

$B$ believes he is following the correct protocol. $Z$ is able to form the correct response in (5) because he knows the compromised key $K'ab$. He can now engage in communication with $B$ using the compromised key and masquerade as $A$. Denning and Sacco suggested that the problem could be fixed by the use of timestamps [42]. The original authors suggested an alternative fix to this problem by means of an extra handshake at the beginning of the protocol [88].

## 4.2 Type Flaws

A message consists of a sequence of components each with some value (for example, the name of a principal, the value of a nonce, or the value of a key). The message is represented at the concrete level as a sequence of bits. A *type flaw* arises when the recipient of a message accepts that message as valid but imposes a different interpretation on the bit sequence than the principal who created it.

For example, consider the Andrew Secure RPC Protocol

$$\begin{array}{lll}
(1)\ A \rightarrow B & : & A, E(Kab : Na) \\
(2)\ B \rightarrow A & : & E(Kab : Na + 1, Nb) \\
(3)\ A \rightarrow B & : & E(Kab : Nb + 1) \\
(4)\ B \rightarrow A & : & E(Kab : K'ab, N'b)
\end{array}$$

Here, principal $A$ indicates to $B$ that he wishes to communicate with him and sends an encrypted nonce $E(Kab : Na)$ as a challenge in (1). $B$ replies to the challenge and issues one of his own by sending the message $E(Kab : Na + 1, Nb)$. $A$ replies to $B's$ challenge by forming and sending $E(Kab : Nb + 1)$ to $B$. $B$ now creates a session key $K'ab$ and distributes it (encrypted) together with a sequence number identifier $N'b$ for future communication.

However, if the nonces and keys are both represented as bit sequences of the same length, say 64 bits, then an intruder could record message (2), intercept message (3) and replay message (2) as message (4). Thus the attack looks like:

$$\begin{array}{lrll}
(1) & A \rightarrow B & : & A, E(Kab : Na) \\
(2) & B \rightarrow A & : & E(Kab : Na + 1, Nb) \\
(3) & A \rightarrow Z(B) & : & E(Kab : Nb + 1) \\
(4) & Z(B) \rightarrow A & : & E(Kab : Na + 1, Nb)
\end{array}$$

Thus principal $A$ may be fooled into accepting the nonce value $Na + 1$ as the new session key. The interpretations imposed on the plaintext bit string of the message are shown in figure 7.

The use of the nonce value as a key may not lead to a security compromise but it should be noted that nonces cannot be assumed to be good keys. Furthermore, nonces do not necessarily have to be random, just unique to the protocol run. Thus a predictable nonce might be used. In such cases $A$ will have been fooled into accepting a key whose value may be known to the intruder.

The above protocol is flawed in other ways too. For example, it is equally possible to record message (4) of a previous run and replay it in the current run, i.e. there is a freshness attack, as pointed out by Burrows, Abadi and Needham [26].

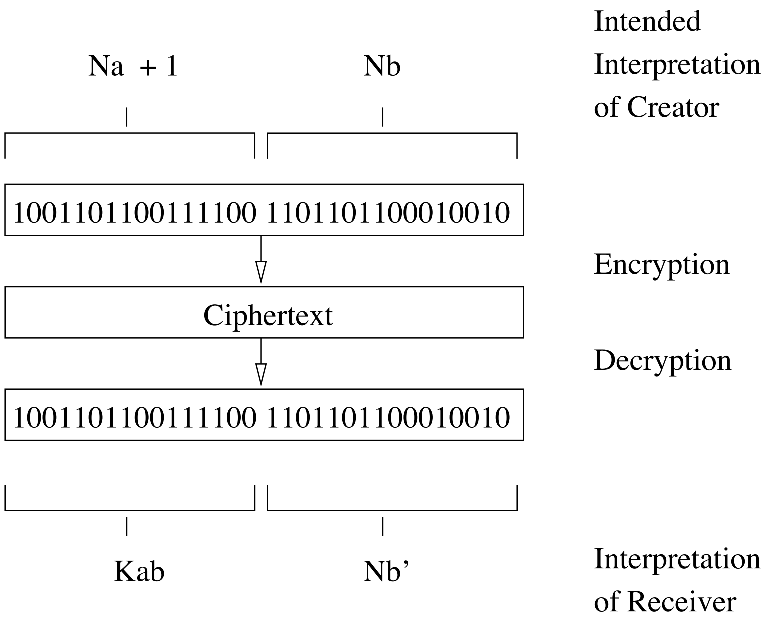


Figure 7: Bit Stream Interpretations and Type Flaw

The Otway-Rees protocol [91] provides another example of a protocol subject to a type flaw attack.

$$\begin{array}{lll}(1)\ A \rightarrow B & : & M, A, B, E(Kas : Na, M, A, B) \\ (2)\ B \rightarrow S & : & M, A, B, E(Kas : Na, M, A, B), E(Kbs : Nb, M, A, B) \\ (3)\ S \rightarrow B & : & M, E(Kas : Na, Kab), E(Kbs : Nb, Kab) \\ (4)\ B \rightarrow A & : & M, E(Kas : Na, Kab)\end{array}$$

The above protocol causes a key *Kab* created by the trusted server *S* to be distributed to principals *A* and *B*. *M* is a protocol run identifier.

After initiating the protocol *A* expects to receive a message back in (4) that contains the nonce *Na* used in (1) together with a new session key *Kab* created by *S*. If *M* is (say) 32 bits long, *A* and *B* each 16 bits long and *Kab* is 64 bits then an intruder *Z* can simply replay the encrypted component of message (1) as the encrypted component of message (4). Thus

$$\begin{array}{lll}(1)\ A \rightarrow Z(B) & : & M, A, B, E(Kas : Na, M, A, B) \\ (4)\ Z(B) \rightarrow A & : & M, E(Kas : Na, M, A, B)\end{array}$$

Here *A* decrypts $E(Kas : Na, M, A, B)$ checks for the presence of the nonce *Na* and accepts (M,A,B) as the new key. M, A and B are all publicly known (since they were broadcast in the clear). Similarly, it is clear that an intruder can play the role of $S$ in messages (3) and (4) simply replaying the encrypted components of message (2) back to $B$. The attack is:

$$\begin{array}{lcl}
(1)\quad A \rightarrow B & : & M, A, B, E(Kas : Na, M, A, B) \\
(2)\ B \rightarrow Z(S) & : & M, A, B, E(Kas : Na, M, A, B), E(Kbs : Nb, M, A, B) \\
(3)\ Z(S) \rightarrow B & : & M, E(Kas : Na, M, A, B), E(Kbs : Nb, M, A, B) \\
(4)\quad B \rightarrow A & : & M, E(Kas : Na, M, A, B)
\end{array}$$

He can now listen in to conversation between $A$ and $B$ using the now publicly available key $(M, A, B)$.

Further examples of type flaws are given by Syverson [109] and Hwang *et al* [60].

## 4.3 Parallel Session Attacks

A parallel session attack occurs when two or more protocol runs are executed concurrently and messages from one are used to form messages in another.

As a simple example consider the following one-way authentication protocol:

$$\begin{array}{lcl}
(1)\ A \rightarrow B & : & E(Kab : Na) \\
(2)\ B \rightarrow A & : & E(Kab : Na + 1)
\end{array}$$

Successful execution should convince $A$ that $B$ is operational since only $B$ could have formed the appropriate response to the challenge issued in message (1). In addition, the nonce $Na$ may be used as a shared secret for the purposes of further communication between the two principals. In fact, an intruder can play the role of $B$ both as responder and initiator. The attack works by starting another protocol run in response to the initial challenge.

$$\begin{array}{lcl}
(1.1)\ A \rightarrow Z(B) & : & E(Kab : Na) \\
(2.1)\ Z(B) \rightarrow A & : & E(Kab : Na) \\
(2.2)\ A \rightarrow Z(B) & : & E(Kab : Na + 1) \\
(1.2)\ Z(B) \rightarrow A & : & E(Kab : Na + 1)
\end{array}$$

Here $A$ initiates the first protocol with message (1.1). $Z$ now pretends to be $B$ and starts the second protocol run with message (2.1), which is simply a replay of message (1.1). $A$ now replies to this challenge with message (2.2). But this is the precise value $A$ expects to receive back in the first protocol run. *Z* therefore replays this as message (1.2). At the very least *A* believes that *B* is operational. In fact, *B* may no longer exist. The attack is illustrated in figure 8. Solid arrows indicate messages of the first protocol run, broken arrows indicate messages of the second protocol run.

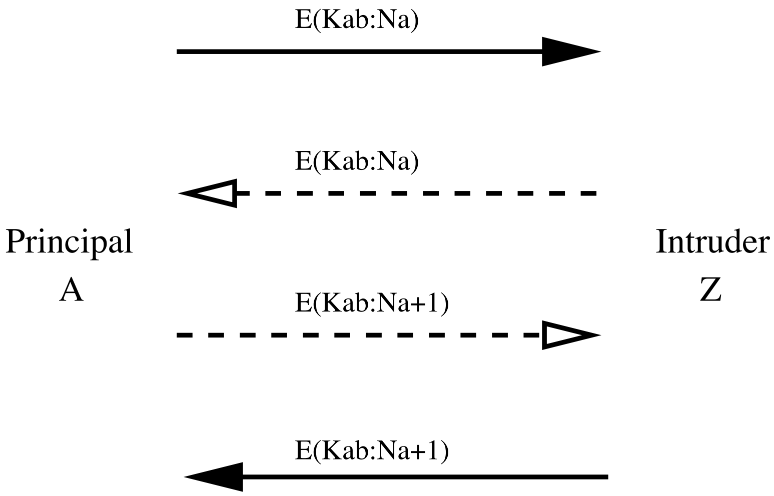


Figure 8: Simple Parallel Session Attack

In the above attack *Z* used principal *A* to do some work on his behalf. He needed to form an appropriate response to the encrypted challenge but could not do so himself and so he "posed the question" to *A* who provided the answer. *A* is is said to act as an **oracle** (because he always provides the correct answer) and attacks of this form are often called oracle attacks.

An interesting example of an oracle attack occurs in the Wide-Mouthed Frog Protocol (not intended for use in real systems). The protocol is described by Burrows, Abadi and Needham [26].

$$\begin{array}{lll} (1)\ A \rightarrow S & : & A, E(Kas : Ta, B, Kab) \\ (2)\ S \rightarrow B & : & E(Kbs : Ts, A, Kab) \end{array}$$

Here, each principal (*A* and *B* in the above) shares a key with the server *S*. If *A* wishes to communicate with a principal *B* then he generates a key *Kab* and a timestamp *Ta* and forms message (1) which is sent to *S*.

On receiving message (1) *S* checks whether the timestamp *Ta* is "timely" and, if so, forwards the key to *B* with its own timestamp *Ts*. *B* checks whether message (2) has a timestamp that is later than any other message it has received from *S* (and so will detect a replay of this message).

The first way it can be attacked is by simply replaying the first message within an appropriate time window - this will succeed since *S* will produce a new second message with an updated timestamp. If $S$'s notion of timeliness is the same as $B$'s (i.e. it accept messages only if the timestamp is later than that of any other message it has received from the sender) then this attack will not work.

The second method of attack allows one protocol run to be recorded and then the attacker continuously uses $S$ as an oracle until he wants to bring about re-authentication between $A$ and $B$.

$$\begin{array}{lrcl}
(1) & A \rightarrow S & : & A, E(Kas : Ta, B, Kab) \\
(2) & S \rightarrow B & : & E(Kbs : Ts, A, Kab) \\
(1') & Z(B) \rightarrow S & : & B, E(Kbs : Ts, A, Kab) \\
(2') & S \rightarrow Z(A) & : & E(Kas : T's, B, Kab) \\
(1'') & Z(A) \rightarrow S & : & A, E(Kas : T's, B, Kab) \\
(2'') & S \rightarrow Z(B) & : & E(Kbs : T''s, A, Kab)
\end{array}$$

$Z$ now continues in the above fashion until he wishes to get $A$ and $B$ to accept the key again. He does this by allowing $A$ and $B$ to receive messages intended for them by $S$.

Parallel session attacks abound in the literature [103, 117, 108, 60]. Bird *et al* [18, 19] illustrate parallel session attacks and present informal methods for analysing for their presence.

## 4.4 Implementation Dependent Attacks

Carlsen [31] indicates that some protocol definitions allow both secure and insecure implementations. Typing attacks could be prevented if the concrete representations of component values contained redundancy to identify a sequence of bits as representing a specific value of a specific type (and the principals made appropriate checks). Few protocol descriptions require such enforcement of types explicitly. Thus, the implementation approach adopted may severely affect the actual security of a protocol that conforms to the description and implementation-dependent attacks are possible.

Similarly we saw in subsection 4.2 how the implementation of nonces (random or predictable) could severely affect the security of a protocol. In that case it merely determined the degree of damage caused by an already flawed protocol.

Perhaps the most interesting (and least understood) area where implementation-dependent attacks may arise is the interaction between a specific protocol and the actual encryption method used. In the protocols we have described so far little has been said about the properties required of an encryption algorithm. The next section shows that the naïve use of certain algorithms (that are generally considered *strong*) in the context of specific protocols may produce insecure results.

### 4.4.1 Stream Ciphers

A stream cipher encrypts a plaintext bit stream on a bit-by-bit basis. The encrypted value of a particular bit may depend on the key $K$, random initialisation data $R$ and the plaintext bits encrypted so far.

Consider the last two messages of the Needham Schroeder protocol described in section 3.2.

$$\begin{array}{lcl} (4)\ B \rightarrow A & : & E(Kab : Nb) \\ (5)\ A \rightarrow B & : & E(Kab : Nb - 1) \end{array}$$

Suppose that the cipherstream for message (4) is $b_1 b_2 \ldots b_{n-1} b_n$. Now if $Nb$ is odd then the final plaintext bit (assumed to be the least significant bit) will be 1 and $Nb - 1$ will differ only in that final bit. On a bit by bit encryption basis, the cipherstream for message (5) can be formed simply by flipping the value of the final bit $b_n$. On average the nonce will be odd half of the time and so this form of attack has a half chance of succeeding. This form of attack was originally described by Boyd [21]. It appears that this form of attack is not limited to stream ciphers. Analysis reveals that similar attacks can also be mounted against certain uses of cipher feedback mode for block ciphers. Furthermore, if the element that is subject to bit flipping represents a timestamp then the scope for mischief seems greater (but seems unrecorded in the literature).

It is interesting to note that under the same set of assumptions a much more virulent attack can be carried out by $A$. Message (3) of the protocol is given below:

$$(3)\ A \rightarrow B \quad : \quad E(Kbs : Kab, A)$$

Flipping the final bit of this message could turn the $A$ into a $C$ under decryption. Since $A$ knows the key $Kab$ he could fool $B$ into believing he shared this key with $C$ and effectively masquerade as $C$.

### 4.4.2 Cipher Block Chaining

Another form of attack concerns the use of Cipher Block Chaining described in section 2.2.3. For any successive pair of ciphertext blocks we can recover the plaintext block corresponding to the second (provided we have the key). Suppose that $E(K : P_1P_2P_3) = IC_1C_2C_3$ Then $C_1C_2C_3$ looks like a ciphertext that begins with initialisation block $C_1$, and decrypts to $P_2P_3$. Similarly $C_1C_2$ decrypts to $P_2$ (it uses $C_1$ as an initialisation block) and $C_2C_3$ decrypts to $P_3$.

Thus we can see that without appropriate additional protection valid messages may be created if their contents are subsequences of generated messages. To distinguish this form of attack from those that follow we shall call this form of flaw a *subsequence flaw*.

Consider again message (2) of the Needham Schroeder protocol of subsection 4.1.

$$(2)\ S \rightarrow A \quad : \quad E(Kas : Na, B, Kab, E(Kbs : Kab, A))$$

Suppose that this has ciphertext $C_0C_1C_2C_3\ldots$ and that all components have length one block. Then $E(Kas : Na, B) = C_0C_1C_2$. But such a message is of the form $A$ might expect to receive in message (3) when $B$ has initiated the protocol. Thus, he can be fooled into accepting the publicly known $Na$ as a key. Thus use of CBC mode of encryption with this protocol will not suffice.

Stubblebine and Gligor [106] have demonstrated attacks via *cut and paste* methods where the ciphertexts of messages are split and conjoined appropriately to form the ciphertexts of other messages (which should only be formable by those in possession of the appropriate key). This is illustrated in figure 9.

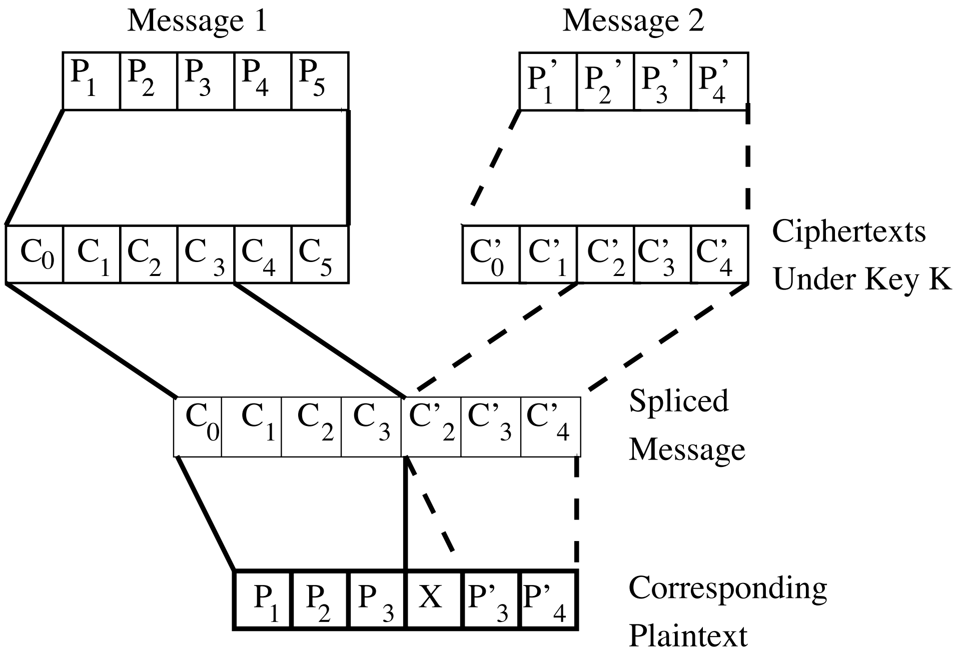
We see that the spliced ciphertext message decrypts to appropriate plaintext except for the block immediately after the join. Denoted by $X$ in the figure, it is likely that it is random gibberish but in some cases that may be precisely what is expected (e.g. if the block is expected to contain a random number). Mao and Boyd have also highlighted the dangers of CBC use [79], pointing out that in many cases it will be possible to determine precisely what value $X$ takes if the intruder has knowledge of the plaintext block corresponding to the ciphertext immediately after the ciphertext join. In the example shown in figure 9, we have

$$X = C_3 \oplus d_K(C_2')$$

$$X = C_3 \oplus (C_1' \oplus P_2')$$

and so if $P_2'$ is known then so is $X$ since the ciphertext blocks are publicly broadcast.

It is dangerous to believe that attacks of the above form lose their power if the plaintext block is not publicly known or guessable; such blocks will


Message 1
Message 2
$P_1$ $P_2$ $P_3$ $P_4$ $P_5$
$P_1'$ $P_2'$ $P_3'$ $P_4'$
$C_0$ $C_1$ $C_2$ $C_3$ $C_4$ $C_5$
$C_0'$ $C_1'$ $C_2'$ $C_3'$ $C_4'$
Ciphertexts
Under Key K
$C_0$ $C_1$ $C_2$ $C_3$ $C'_2$ $C'_3$ $C'_4$
Spliced
Message
$P_1$ $P_2$ $P_3$ X $P'_3$ $P'_4$
Corresponding
Plaintext

$$X = C_3 \oplus d_K(C'_2)$$

Figure 9: Splicing Attack on Cipher Block Chaining

generally be known to the parties communicating in a protocol who may misuse their knowledge (see below).

Of particular note are *initialisation attacks* — attacks that involve modulation of the initialisation vector $C_0$. Consider a ciphertext that starts with $C_0C_1$ and suppose that we know that the initial plaintext block was $P_1$. Then

$$P_1 = C_0 \oplus d_K(C_1)$$

Now for any desired block value $W$ we have

$$W = W \oplus P_1 \oplus P_1$$

since anything XORed with itself is 0. And so we have (substituting for the second $P_1$)

$$W = W \oplus P_1 \oplus (C_0 \oplus d_K(C_1))$$

and so

$$W = C_0' \oplus d_K(C_1)$$

where $C_0' = W \oplus P_1 \oplus C_0$ and so $C_0'C_1$ is the ciphertext corresponding to plaintext $W$. In this fashion we can replace the initial known plaintext block $P_1$ with our own choice $W$. This is potentially very disturbing since the rest of the message is unaffected.

As an example of the danger of this attack, consider again message (2) of the Needham Schroeder protocol. We can record message (2) of a previous run of this protocol between $A$ and $B$. In particular we can replay the old message (2) after modifying the initial block from the old (and known) value of the nonce $Na$ with the new one issued in the current run of the protocol. Thus, we can impersonate the trusted server $S$. Now consider the contents of message (3) of that protocol:

$$(3)\ A \rightarrow B \quad : \quad E(Kbs : Kab, A)$$

Since $A$ knows the key in message (3), he can create a new message (3) whenever he likes for any key value he likes. One might argue that if $A$ wants to misbehave he can do so much more simply than this but this misses the point: $B$ works on the assumption that the contents of message (3) were created by the trusted server $S$. This is clearly not the case.

We have illustrated these attacks using the Needham Schroeder protocol simply because it is the best known and simple to understand. The above forms of attack present problems with a good number of protocols.

We have illustrated various forms of cryptoalgorithm dependent flaws. The above description is by no means exhaustive. Indeed, other modes of encryption have given rise to problems in implemented protocols. In particular, Propagating Cipher Block Chaining (PCBC) mode was shown to be deficient and led to the Kerberos V.5 protocol adopting CBC mode (V.4 used PCBC). Criticisms of the Kerberos protocols were given by Bellovin and Merritt [14]. Other aspects relating to Cipher Block Chaining can be found in the recent paper by Bellare *et al* [13].

## 4.5 Binding Attacks

In public key cryptography the integrity of the public keys is paramount. Suppose your public key is $Ky$ and an intruder's public key is $Ki$. The intruder is able to decrypt any messages encrypted with $Ki$. Principals wishing to convey information to you secretly will encrypt using what they believe is your public key. Thus, if the intruder can convince others that your public key is $Ki$ then they will encrypt secret information using $Ki$ and this will be readable by the intruder.

Thus, the principals in charge of distributing public keys must ensure that the above cannot occur; there must be a verifiable binding between a public key and the corresponding agent. In some authentication protocols, this has not been achieved. Consider the following protocol:

(1) $C \rightarrow AS \quad : \quad C, S, Nc$
(2) $AS \rightarrow C \quad : \quad AS, E(Kas^{-1} : AS, C, Nc, Ks)$

Here, a prospective client $C$ wants to communicate with $S$ and needs the public key of $S$. The *certification authority* $AS$ is the repository for principals' public keys. $C$ sends message (1) to request the public key of $S$. He includes a nonce $Nc$ to ensure the freshness of the expected reply.

$AS$ replies with message (2). The principal identifier $AS$ is sent in the clear to tell $C$ which public key to use to decrypt the following ciphertext. The components of the encrypted part signify that the message was created by $AS$, that this message has been created in response to a request from a client $C$ with nonce $Nc$ and that the public key requested is $Ks$. However, the reader may note that there is nothing in the encrypted part of message (2) that assures the recipient that the key is really the public key of $S$. This leads to the following attack:

(1.1) $C \rightarrow Z(AS) \quad : \quad C, S, Nc$
(2.1) $Z(C) \rightarrow AS \quad : \quad C, Z, Nc$
(2.2) $AS \rightarrow Z(C) \quad : \quad AS, E(Kas^{-1} : AS, C, Nc, Kz)$
(1.2) $Z(AS) \rightarrow C \quad : \quad AS, E(Kas^{-1} : AS, C, Nc, Kz)$

Here the intruder $Z$ intercepts the initial message from $C$ to $AS$ and simply replaces the identifier of the intended server $S$ with his own identifier $Z$. and sends the result to $AS$ as message 2.1. $AS$, believing that $C$ has requested $Z$'s public key, replies with message (2.2). $Z$ simply allows this to be received by $C$ as message (1.2). $C$ performs appropriate decryptions and checks and believes that he has received the public key of $S$. This attack (and a similar one) was identified by Hwang and Chen [59]. They suggest that this problem can be solved by explicitly including the identifier of the requested server $S$ in message (2). The protocol then becomes:

$$\begin{array}{lcl} (1)\ C \rightarrow AS & : & C, S, Nc \\ (2)\ AS \rightarrow C & : & AS, E(Kas^{-1} : AS, C, Nc, S, Ks) \end{array}$$

Problems with signing after encryption arose some time ago with the draft CCITT X.509 standard. L'Anson and Mitchell [12] showed certain deficiencies in the protocols as did Burrows, Abadi and Needham [26] (see 6.9.1).

## 4.6 Encapsulation Attacks

In a great many protocols a principal $A$ may arrange for a second principal $B$ to encrypt some data chosen by $A$. As a rule such data should be regarded as 'user data' and carefully considered as a vehicle for cryptosystem dependent attacks. As a simple example consider the following key translation protocol due to Davis and Swick [40]:

$$\begin{array}{lcl} (1)\ B \rightarrow A & : & E(Kbt : A, msg) \\ (2)\ A \rightarrow T & : & E(Kbt : A, msg), B \\ (3)\ T \rightarrow A & : & E(Kat : msg, B) \end{array}$$

In this protocol all participants share keys with the trusted server $T$. The server acts as intermediary. $A$ accepts message (3) as proof that $B$ sent the message *msg* to him via $T$. The reversal of principal and message components appears to be made to introduce asymmetry (and hence protect against reflections). However, if *msg* begins with a principal identifier $C$ then message (3) may be passed off as a message (1) but originated by $A$ and intended for $C$. Since $B$ chooses the contents of *msg* he can arrange this. Can we protect against this by means of some integrity check? Generally the answer will be yes but this is not without its pitfalls. If messages are of variable length then in many cases, it may be possible to embed a whole message (including CRC check say) in the *msg* component. If CBC mode of encryption is used then a perfectly formed encrypted message could be extracted (this depends on how initial vectors are chosen).

Note that user data is very common, typically in the form of principal identifiers or nonces and the like. Thus if a plaintext message $P$ were 3 blocks long (including checks) and another message had a freely chosen nonce $N$ then if this nonce is 3 or more blocks in length then a CBC encapsulation attack becomes possible. Similar attacks will hold when a stream cipher is used (but here it will generally have to be the initial segment of a message).

## 4.7 Other Forms of Attack

The above forms of attack may be regarded as representative of the dangers involved in designing authentication protocols. In general, they do not require a great deal of mathematical sophistication to comprehend. More sophisticated attacks that take advantage of particular algebraic properties of the cryptoalgorithm when used in the context of authentication protocols are given in the excellent paper by Judy Moore [85].

In addition, more traditional forms of attack such as cryptanalysis can be launched on several protocols. Mao and Boyd [80] have recently investigated ways of protecting against such attacks. Paul Kocher's recent discovery of an attack on RSA via timing analysis might well have profound and more general impact [72].

Aspects of redundancy have also been addressed by Gong [51]. Protocols using passwords have been addressed by several authors [15], [54]. Carlsen [31] has a category called *elementary flaws* which is used to group protocols which are breakable with little effort because they provide little or no protection. The (unintentionally flawed) CCITT X.509 protocol and the (intentionally flawed — it was intended as a example to highlight deficiencies in the use of BAN logic) Nessett protocol [89] are included in this category. It is a matter of opinion as to when a flaw is considered elementary and the choice is somewhat arbitrary. Clark and Jacob have discovered a flaw similar to the CCITT X.509 one in a recently published protocol [34]. Anderson and Needham provide introductory accounts of how protocols may fail and provide good advice on how to construct secure protocols [10, 11]. Anderson also provides a highly readable and somewhat distressing account of how management aspects as well as technical aspects can cause systems to fail [5].

## 4.8 Conclusions

Protocol construction might seem a simple task; protocols often comprise only a few messages. This is, however, clearly deceptive and the examples we have shown above indicate that the invention of secure protocols is a remarkably subtle affair.

The current explosion in distributed system development and network usage means that there is a pressing need for a framework and tool support for the rigorous development and analysis of new security protocols. Although significant advances have been made in recent years, there is clearly some way to go! As Lowe has shown [76] the same mistakes seem to be made time after time.

There are, however, signs that the community is getting to grips with the matter at hand. There is a gradual realisation that it is the whole system that is important and that a considerable number of factors need to be taken into account. Anderson emphasises the management aspects in banks [5]. Abadi and Needham take a strong practical engineering approach providing ten useful rules of thumb in their excellent general design guide [1].

The subtlety of some attacks indicates that a systematic (and automated) approach to analysis is essential. The next section indicates some of the methods and tools that have been used to date.

# 5 Formal Methods for Analysis

In this section we review the major approaches to the specification and analysis of authentication protocols. Several methods have been tried, each with their strengths and weaknesses. We address them as follows:

- the use of existing formal methods to specify and analyse authentication protocols;
- the use of logics of knowledge and belief;
- the use of expert systems and algebraic term-rewriting systems.

This is the classification used by Rubin and Honeyman [95] in their review article. As indicated by Rubin and Honeyman, the above methods as implemented are all independent of the cryptographic mechanism used. This is of course a strength since in producing a protocol specification we might not yet wish to specify a particular implementational mechanism. However, it also highlights a gap in the formal support for protocol development: tool support for the identification of cryptosystem dependent insecurities.

## 5.1 Extant Formal Verification Systems

Early formal efforts concentrated on the use of existing formal specification and verification systems. This is hardly surprising; a great deal of effort was expended by the security community (developers, evaluators and Government agencies) to use formal specification and verification techniques for many aspects of security. Toolsets had been developed, or were being developed, and use could be made of the experience gained in other areas.

The first such attempt appears to be that of Kemmerer, who in 1987 used the Ina Jo development environment to specify and prove properties of a cryptographic system [69]. The attempt was successful and demonstrated that proof technology could be brought to bear successfully on problems in the field. Effectively, security of the system is expressed as state invariants which are then shown to be maintained under controlled transitions. The work was concerned with the *correctness* of the system. There was no attempt to model, for example, an active intruder on the network.

Boyd and Mao have used the Z specification language to specify aspects of a key distribution systems [24]. No proofs are attempted. The Z language has been used in many areas of security outside of authentication. Within the UK it is often the language of choice for specification for Governmental agencies. Details of these uses are omitted here. More recently, the B notation has been used to specify authentication systems [17]. This method shows some promise as tool support emerges.

The use of such state-based techniques seems of limited use. There is little or no attempt to model an attacker (Kemmerer models a passive intruder, Mao and Boyd model none). There is an implicit assumption that the functionality specified is sufficient to maintain security. Without an explicit statement of what attacks are possible it is impossible to see whether the specified operations actually do maintain security. Such methods are primarily concerned with the preservation of *correctness* rather than security.

Other formal specification techniques have been used for authentication protocols, e.g. LOTOS has been used to specify the X.509 directory framework. Finite-state machines have also been used by several authors for the specification and analysis of protocols. None of these uses provides analytical support for security in the face of an active intruder. Rubin and Honeyman [95] provides some details.

Recent work by the Formal Systems Europe and Programming Research Group at Oxford [94] has used verification techniques for process algebras to analyse security protocols. In particular work has been carried out using CSP. Principals in the protocol are specified as CSP processes operating in parallel. In addition, a general attacker is added that can carry out actions that may reasonably be expected of an attack (listening, faking, replaying etc.)

An authentic run of the protocol is specified (the protocol terminates with success only if the message sequence is what the protocol intended). The implementation of the protocol which comprises the various principals as agents must now be shown to satisfy the specification. The Failures Divergences Refinement (FDR) tool is used to check possible traces of the implementation against the specification. Roscoe and Gardiner have created a variety of heuristics to prune down the search space to make the model checking feasible.

The results have been very promising (subtle and hitherto unknown protocol flaws have been discovered using the approach). For example, 17 years after its publication a flaw was found in the Needham Schroeder Public Key protocol by Lowe using the FDR tool [75]. See also [77]. Roscoe and Gardiner provide an account of the initial results of their research in [94]. The extension of the work to handle algebraic elements is also available [47] [48]. A particularly pleasing part of the work is the willingness to investigate the operation of protocols under the relaxation of trust in principals (or the weakening of assumptions). Further work on the use of CSP to capture and verify security properties has been carried out by Schneider [97]. An overview of the CSP approach to authentication protocols is given by Ryan [96].

## 5.2 The Use of Logics

Logics have seen widespread use in the analysis of authentication protocols. The logics used have been principally of two types:

- *epistemic* logics (that is, logics of knowledge);
- *doxastic* logics (that is, logics of belief).

Traditionally, issues of trust have been dealt with using belief logics and issues of security have been dealt with using knowledge logics. Syverson [107] provides a good overview of how logics can be used for the analysis of authentication protocols. He indicates that it is possible to reason about both trust and security using either approach but that in practice he has found that epistemic logics are more efficient. The greatest amount of effort has been expended in the use of belief logics and it is to this that we turn our attention first.

### 5.2.1 BAN Logic

In 1989, Burrows, Abadi and Needham published what is probably the most influential document in authentication literature [26]. They provided a logic (referred to universally as *BAN logic*) to describe the beliefs of principals involved in a protocol. The set of beliefs held by a principal changes as he receives protocol messages. The authors provide a set of inference rules that define how the set of beliefs changes. Thus, given an initial set of beliefs the logic allows the analyst to determine what the final belief state is.

BAN logic has a special place in authentication history; it represents the first attempt to provide a formal language to describe what the *assumptions* of a protocol are and also what the *goals* are. In general, protocol descriptions have generally stated *what* the principals should *do* and not what they were trying to *achieve*.

The logic has stimulated a great deal of controversy. Nessett [89] provides an example of a clearly insecure protocol which is nevertheless accepted as secure by the BAN logic. Effectively, a shared key *K* is encrypted under a private key and broadcast to the network. Since the corresponding public key is generally known, the message can be decrypted by all to obtain the secret shared key. In their rejoinder [27] the BAN authors point out that their logic dealt with *trust* and not confidentiality, stating that the obvious publication of the shared key in the indicated manner contradicted a belief in its suitability for use.

This would appear correct, but the situation is still rather unsatisfying. Additional problems have been identified. Snekkenes [102] showed that permutations of protocol steps left the results unaffected.

It is possible that a principal may decrypt some random text to obtain some putative ”formula” using some key that he holds. For the decryption to succeed the result of decryption must be *meaningful* in some way. Gong, Needham and Yahalom [55] introduce the notion of recognisability in their logic (general referred to as GNY) to cater for this. Also, the original BAN logic assumes that there is sufficient redundancy in a message for a principal to detect a message he himself originated (thus reflection attacks are assumed to be catered for outside of BAN analysis). GNY logic makes origination explicit. GNY allows preconditions to be attached to rules to achieve different levels of belief. Thus, different levels of trust are allowed by the logic. Most BAN work concentrates on the *analysis* of protocols. When used for development, problems may arise because completely infeasible protocols may be specified that nevertheless achieve the desired goals according to the protocol (e.g. by specifying that principals send messages that contain information they simply do not have). This is dealt with by Gong [52] whose extended logic requires that principals make use only of information that is legitimately available to them.

Boyd and Mao [24] provide many criticisms of BAN logic (and other descendants): the formalisation approach is somewhat vague; it allows beliefs that may legitimately be regarded as nonsensical (e.g. belief in a nonce) and the method of determining assumptions is ad hoc. Instead they provide a language for describing protocols and a partially mechanised approach to idealisation. As pointed out by Rubin and Honeyman [95] there is still informal judgement at work in the idealisation process. The reasoning process is backwards (rather than forwards as in BAN logic), thus the reasoning proceeds from the desired conclusion to derive initial beliefs.

There have been other belief-logic approaches. Boyd and Mao have introduced a non-monotonic logic of belief (i.e. one which allows previously held beliefs to be revoked) [24]. Campbell *et al* [30] introduce the notion of uncertainty into BAN by assigning probabilities to assumptions and to rules of inference. This allows conclusions drawn to be treated as uncertain. Linear programming methods are used to determine the precise bounds of probabilities.

Kessler and Wedel modify BAN to allow the incorporation of plaintext messages [71]. This widens the scope of what can be analysed. They also replace the nonce-verification rule of BAN with a ”has recently said ” rule. A recipient of a message no longer believes that the sender of a message believes the contents, rather he now just deduces that the sender sent it recently. A rule is introduced to allow a principal to try keys that he has (or can generate) without actually believing that the key is appropriate for the message in question. Kessler and Wedel’s most important suggestion is the incorporation of a passive eavesdropper into the system. By the determination (by closure) of information available to such an intruder, certain types of confidentiality breaches can be detected (e.g. the the Nessett flaw). The authors provide an example of BAN’s inability to deal with a parallel session attack. Recent work by Boyd and Mao has indicated that care needs to be taken when cleartext is omitted [23] but Oorschot disputes the views they take [120].

Overall, BAN has proved of substantial use. It often seems like a marked improvement on its successors which have added conceptual apparatus to deal with its perceived deficiencies at the expense of considerable increase in complexity. This is indeed the view of Roger Needham (commenting on GNY logic). Kessler and Wedel note that BAN extensions tend to be extensions to the original BAN logic, not to its successors. BAN logic has unearthed many protocol flaws and provides a very cost-effective means of detecting (some) flaws. In terms of value for money it has much to be said for it. The rule would appear to be ”Try BAN first; it doesn’t cost a great deal and it often produces results.” The method is clearly not without its difficulties; it should be regarded as a useful tool. BAN logic deals with with the beliefs of trustworthy principals; it does not deal with confidentiality. Since confidentiality is essential to maintaining authentication other methods will need to be brought to bear for system security.

An important aspect of the BAN approach is that it forces the analyst to be precise about what the goals and assumptions of a protocol actually are. It is often very difficult to determine these from many specifications.

### 5.2.2 Other Logics

General purpose logics (or adapted forms) have also been employed in the services of authentication. Bieber [16] provides a quantified extension called CKT5 of the modal logic KT5 (together with send and receive operators) and uses it to couch and prove authentication properties. Carlsen

[32] indicates how various deficiencies of the standard notation can be overcome by providing rules for a standard protocol specification into a CKT5 logic specification. Snekkenes has shown that the sort of analysis carried out in the Bieber method is insufficient to detect *multi-role flaws*, i.e. where a principal does not restrict himself to playing just one agent. He also suggests how to extend Bieber's approach to cope with the problem. Snekkenes notes in his doctoral dissertation that principal operation is couched in rather complex formulae. Snekkenes has also carried out significant work that uses the HOL (Higher Order Logic) specification language and tool support to specify and prove properties about protocols [104].

## 5.3 Expert Systems and Algebraic Rewriting Systems

There have been a few notable attempts to provide automated analysis of protocols via search techniques. Early work by Millen *et al* led to the development of the Interrogator tool [84]. The user guide provides an updated account of the tools facilities [83]. Protocols are specified in a prolog-based syntax. Knowledge of the various principals is built up and recorded as the protocol progresses. The tool, with guidance from the user, can be used to investigate ways in which states can be reached where security is compromised, i.e. start from an insecure state and attempt to see how you could have got there. The tool appears usable and has been used to find flaws in protocols. It is one of the tools included in a comparative study of three systems [70]. The comments there indicate that the tool at present has problems in discovering flaws in which a principal takes on more than one role (if so this is a weakness shared with other systems, see [103]). Also the paper notes

> There are, in general, many different ways to specify the same protocol, which are "correct" in some sense. Yet they lead to different running times, and some may exclude possible penetrations.

Search-path pruning heuristics may lead to some penetrations being missed. Snekkenes [104] points out that the Interrogator does not allow the identification of guess-based attacks. BAN logic does not address these either. As far as we are aware the Interrogator has not discovered any new attacks.

Meadows has developed an analysis tool based on term rewriting (the NRL Protocol Analyser). The specification language is again prolog-based and fairly easy to follow. Principals possess beliefs and also know various words which make up messages. Receipt of a message causes the state of the system to change. Words and beliefs held by a principal occur as a result of receiving messages. Various rewrite rules are specified as part of the protocol (e.g. the result of encrypting and then decrypting some plain text with the same key produces te original plaintext). The tool attempts to find scenarios to reach an insecure state. The tool looks technically effective but Rubin and Honeyman [95] report that these types of tools are rather difficult to use by designers. Interestingly, the tool failed to find a flaw in the TMN protocol due to the way in which the properties of the RSA algorithm had been couched [70]. The analysis process is not entirely automated; lemmas for the tool to prove must be generated by the user.

# 6 A Library of Protocols

## 6.1 Symmetric Key Protocols Without Trusted Third Party

### 6.1.1 ISO Symmetric Key One-Pass Unilateral Authentication Protocol

This protocol [62] consists of a single message from one principal $A$ to a second $B$. A secret key $Kab$ is assumed to be shared between these two principals.

$$(1)\ A \rightarrow B \quad : \quad Text2, E(Kab : [Ta|Na], B, Text1)$$

The use of the text fields is application specific. There is a choice between a sequence number Na and a timestamp Ta which 'depends on the technical capabilities of the claimant and the verifier as well as the environment.'

### 6.1.2 ISO Symmetric Key Two-Pass Unilateral Authentication Protocol

In this protocol the claimant $A$ is authenticated by the verifier $B$ by the means of challenge-response. The protocol is fairly familiar:

$$\begin{array}{lll}(1)\ B \rightarrow A & : & Rb, Text1 \\ (2)\ A \rightarrow B & : & Text3, E(Kab : Rb, B, Text2)\end{array}$$

Here $Rb$ is a random number. On receiving message (2) $B$ decrypts the encrypted component and checks for the presence of both $B$ and $Rb$ issued in message (1). At the end of the protocol $B$ may conclude that $A$ is operational (or at least was the originator of message (2) after he ($B$) issued message (1)).

### 6.1.3 ISO Symmetric Key Two-Pass Mutual Authentication

This protocol allows each communicating principal to establish that the other is operational. Again, a secret key is assumed to be shared between $A$ and $B$.

$$\begin{array}{lll}(1)\ A \rightarrow B & : & Text2, E(Kab : [Ta|Na], B, Text1) \\ (2)\ B \rightarrow A & : & Text4, E(Kab : [Tb|Nb], A, Text3)\end{array}$$

This protocol is in fact two independent uses of the one-pass authentication protocol (see 6.1.1). Use of the text fields is suggested as a way of binding the two messages.

Again, the use of sequence numbers or timestamps 'depends on the technical capabilities of the claimant and the verifier as well as the environment.'

### 6.1.4 ISO Symmetric Key Three-Pass Mutual Authentication

Here mutual authentication is achieved by the use of random numbers $Ra$ and $Rb$.

$$\begin{array}{lll}
(1)\ B \rightarrow A & : & Rb, Text1 \\
(2)\ A \rightarrow B & : & Text3, E(Kab : Ra, Rb, B, Text2) \\
(2)\ B \rightarrow A & : & Text5, E(Kab : Rb, Ra, Text4)
\end{array}$$

On receiving message (2) $B$ checks for the presence of both $B$ and $Rb$ sent in message (1). On receiving message (3) $A$ checks both $Rb$ and $Ra$ are the ones sent in message (1) and (2) respectively.

### 6.1.5 Using Non-Reversible Functions

In this protocol, the responding principal is trusted to generate a new session key $K$. On receiving message (2) $B$ decrypts and then checks that the correct value of $f(Rb)$ has been sent. He forms a one-way hash value of the other nonce $Ra$ and encrypts it under the newly distributed key $K$ and sends the result to $A$, who similarly decrypts and checks the value is correct.

$$\begin{array}{lll}
(1)\ B \rightarrow A & : & B, Rb \\
(2)\ A \rightarrow B & : & A, E(Kab : f(Rb), Ra, A, K) \\
(3)\ B \rightarrow A & : & B, E(K : f(Ra))
\end{array}$$

### 6.1.6 Andrew Secure RPC Protocol

This protocol has been shown to be flawed. It is intended to distribute a new session key between two principals $A$ and $B$. In the final message (4) the nonce $N'b$ is a handshake number to be used in future messages.

$$\begin{array}{lll}
(1)\ A \rightarrow B & : & A, E(Kab : Na) \\
(2)\ B \rightarrow A & : & E(Kab : Na + 1, Nb) \\
(3)\ A \rightarrow B & : & E(Kab : Nb + 1) \\
(4)\ B \rightarrow A & : & E(Kab : K'ab, N'b)
\end{array}$$

The problem with this protocol is that there is nothing in message (4) that $A$ knows to be fresh. An intruder can simply replay this message at a later date to get $A$ to accept it as the final message of a protocol run (i.e. replace the final message sent by $B$). There is also a parallel session attack.

## 6.2 Authentication Using Cryptographic Check Functions

All ISO protocols in this section can be found in Part 4 of the ISO 9798 Standard [64]. The keyed function $f_{Kab}(X)$ returns a hashed value for data $X$ in a manner determined by the key $Kab$.

### 6.2.1 ISO One-Pass Unilateral Authentication with CCFs

$$\begin{array}{lcl} (1)\ A \rightarrow B & : & [Ta|Na], B, Text2, f_{Kab}([Ta|Na], B, Text1) \end{array}$$

### 6.2.2 ISO Two-Pass Unilateral Authentication with CCFs

$$\begin{array}{lcl} (1)\ B \rightarrow A & : & Rb, Text1 \\ (2)\ A \rightarrow B & : & Text3, f_{Kab}(Rb, B, Text2) \end{array}$$

### 6.2.3 ISO Two-Pass Mutual Authentication with CCFs

$$\begin{array}{lcl} (1)\ A \rightarrow B & : & [Ta|Na], Text2, f_{Kab}([Ta|Na], B, Text1) \\ (2)\ B \rightarrow A & : & [Tb|Nb], Text4, f_{Kab}([Tb|Nb], A, Text3) \end{array}$$

This protocol is two independent uses of the single pass unilateral authentication protocol.

### 6.2.4 ISO Three-Pass Mutual Authentication with CCFs

$$\begin{array}{lcl} (1)\ B \rightarrow A & : & Rb, Text1 \\ (2)\ A \rightarrow B & : & Ra, Text3, f_{Kab}(Ra, Rb, B, Text2) \\ (3)\ B \rightarrow A & : & Text5, f_{Kab}(Ra, Ra, Text4) \end{array}$$

## 6.3 Symmetric Key Protocols Involving Trusted Third Parties

### 6.3.1 Needham Schroeder Protocol with Conventional Keys

This is the most celebrated (or best-known) of all Security Protocols. The original presentation is given in [87] (where it forms *protocol 1*). The more usual notational conventions are adopted here.

$$\begin{array}{lcl} (1)\ A \rightarrow S & : & A, B, Na \\ (2)\ S \rightarrow A & : & E(Kas : Na, B, Kab, E(Kbs : Kab, A)) \\ (3)\ A \rightarrow B & : & E(Kbs : Kab, A) \\ (4)\ B \rightarrow A & : & E(Kab : Nb) \\ (5)\ A \rightarrow B & : & E(Kab : Nb - 1) \end{array}$$

The most famous attack is by Denning and Sacco [42]. There is another potential weakness which depends on the nature of the assumptions made about cryptographic support.

The main problem with this protocol is that $B$ has no way of ensuring that the message (3) is fresh. An intruder can compromise a key and then replay the appropriate message (3) to $B$ and then complete the protocol. Also, $A$ could (should he so wish) also spoof message (3) in the same way (causing a stale key to be reaccepted by $B$).

If a stream cipher is used then the difference between the ciphertexts in (4) and (5) is very small (one bit) and this allows a simple attack to be launched. The reader is referred to [21]. See also section 4.4.1.

### 6.3.2 Denning Sacco Protocol

Denning and Sacco suggested fixing the freshness flaw in the Needham Schroeder protocol above by the use of timestamps. The protocol becomes:

(1) $A \rightarrow S \;:\; A, B$
(2) $S \rightarrow A \;:\; E(Kas : B, Kab, T, E(Kbs : A, Kab, T))$
(3) $A \rightarrow B \;:\; E(Kbs : A, Kab, T)$

$T$ is a timestamp. $B$ can check for timeliness of message (3) (allowing for clock drift and network delays). There is now no need for the extra exchange between $A$ and $B$ of the Needham-Schroeder protocol.

### 6.3.3 Otway-Rees Protocol

The Otway-Rees Protocol [91] is a well-known protocol that has been shown to be flawed. The notation of the original differs from common usage and so the form presented here is that given in [26].

(1) $A \rightarrow B \;:\; M, A, B, E(Kas : Na, M, A, B)$
(2) $B \rightarrow S \;:\; M, A, B, E(Kas : Na, M, A, B), E(Kbs : Nb, M, A, B)$
(3) $S \rightarrow B \;:\; M, E(Kas : Na, Kab), E(Kbs : Nb, Kab)$
(4) $B \rightarrow A \;:\; M, E(Kas : Na, Kab)$

In the above $M$ is a nonce (a run identifier). In message (1) $A$ sends to $B$ the plaintext $M, A, B$ and an encrypted message readable only by the server $S$ of the form shown. $B$ forwards the message to $S$ together with a similar encrypted component. The server $S$ decrypts the message components and checks that the components $M, A, B$ are the same in both messages. If so, then it generates a key $Kab$ and sends message (3) to $B$ which forwards part of the message to $A$. $A$ and $B$ will use the key $Kab$ only if the message components generated by the server $S$ contain the correct nonces $Na$ and $Nb$ respectively.

An attack on the protocol is given below:

$$\begin{array}{lll} (1)\ A \rightarrow Z(B) & : & M, A, B, E(Kas : Na, M, A, B) \\ (4)\ Z(B) \rightarrow A & : & M, E(Kas : Na, M, A, B) \end{array}$$

In this attack principal $A$ is fooled into believing that the triple $M, A, B$ is in fact the new key. This triple is of course public knowledge. This is an example of a *type flaw*. Of course, it is also possible to wait until message (2) of the original protocol has been sent and then reflect appropriate components back to both $A$ and $B$ and then monitor the conversation between them.

### 6.3.4 Amended Needham Schroeder Protocol

In 1987 Needham and Schroeder [88] suggested a fix to the original Needham Schroeder Protocol. This is given below.

$$\begin{array}{lll} (1)\ A \rightarrow B & : & A \\ (2)\ B \rightarrow A & : & E(Kbs : A, Nb^0) \\ (3)\ A \rightarrow S & : & A, B, Na, E(Kbs : A, Nb^0) \\ (4)\ S \rightarrow A & : & E(Kas : Na, B, Kab, E(Kbs : Kab, Nb^0, A)) \\ (5)\ A \rightarrow B & : & E(Kbs : Kab, Nb^0, A) \\ (6)\ B \rightarrow A & : & E(Kab : Nb) \\ (7)\ A \rightarrow B & : & E(Kab : Nb - 1) \end{array}$$

The protocol is thought to be secure (there would appear to be a cryptographic implementation dependent flaw; namely the bit flipping flaw described by Boyd [21]).

### 6.3.5 Wide Mouthed Frog Protocol

The following protocol is given in [26]. It is due to Burrows.

$$\begin{array}{lll} (1)\ A \rightarrow S & : & A, E(Kas : Ta, B, Kab) \\ (2)\ S \rightarrow B & : & E(Kbs : Ts, A, Kab) \end{array}$$

$A$ is trusted to generate a session key $Kab$. On receiving message (1) $S$ checks whether the timestamp $Ta$ is "timely" and, if so, forwards the key to $B$ with its own timestamp $Ts$. $B$ checks whether the message (2) has a timestamp that is later than any other message it has received from $S$. The protocol is flawed (possibly in several ways).

The first way it can be attacked is by simply replaying the first message within an appropriate time window - this will cause re-authentication since $S$ will produce a new second message with an updated timestamp. The second method of attack allows one session to be recorded and then the attacker continuously uses $S$ as an oracle until he wants to bring about re-authentication between $A$ and $B$.

$$\begin{array}{lrcl}
(1) & A \rightarrow S & : & A, E(Kas : Ta, B, Kab) \\
(2) & S \rightarrow B & : & E(Kbs : Ts, A, Kab) \\
(1') & Z(B) \rightarrow S & : & B, E(Kbs : Ts, A, Kab) \\
(2') & S \rightarrow Z(A) & : & E(Kas : T's, B, Kab) \\
(1'') & Z(A) \rightarrow S & : & A, E(Kas : T's, B, Kab) \\
(2'') & S \rightarrow Z(B) & : & E(Kbs : T''s, A, Kab)
\end{array}$$

and now Z is in a position to replay appropriate messages to $A$ and $B$

$$\begin{array}{lrcl}
(1) & A \rightarrow Z(S) & : & E(Kas : T's, B, Kab) \\
(2) & Z(S) \rightarrow B & : & E(Kbs : T''s, A, Kab)
\end{array}$$

The above has been discovered independently by several authors.

### 6.3.6 Yahalom

The Yahalom protocol is given below. It has been shown to be flawed by several authors. There are also some attacks based on assumptions about cryptographic implementation which were noticed by Clark and Jacob (many protocols are equally susceptible).

$$\begin{array}{lrcl}
(1) & A \rightarrow B & : & A, Na \\
(2) & B \rightarrow S & : & B, E(Kbs : A, Na, Nb) \\
(3) & S \rightarrow A & : & E(Kas : B, Kab, Na, Nb), E(Kbs : A, Kab) \\
(4) & A \rightarrow B & : & E(Kbs : A, Kab), E(Kab : Nb)
\end{array}$$

One attack on the Yahalom protocol is given below:

$$\begin{array}{lrcl}
(1) & Z(A) \rightarrow B & : & A, Na \\
(2) & B \rightarrow Z(S) & : & B, E(Kbs : A, Na, Nb) \\
(3) & \rightarrow & : & Omitted \\
(4) & Z(A) \rightarrow B & : & E(Kbs : A, Na, Nb), E(Na, Nb : Nb)
\end{array}$$

Other attacks can be mounted on the protocol. Attacks on a modified form of this protocol can be found in [108].

### 6.3.7 Carlsen's Secret Key Initiator Protocol

This protocol is fairly self-explanatory and may be found in [33].

$$
\begin{array}{lcl}
(1)\ A \to B & : & A, Na \\
(2)\ B \to S & : & A, Na, B, Nb \\
(3)\ S \to B & : & E(Kbs : Kab, Nb, A), E(Kas : Na, B, Kab) \\
(4)\ B \to A & : & E(Kas : Na, B, Kab), E(Kab : Na), N'b \\
(5)\ A \to B & : & E(Kab : N'b)
\end{array}
$$

### 6.3.8 ISO Four-Pass Authentication Protocol

$$
\begin{array}{lcl}
(1)\ A \to B & : & TVPa, B, Text1 \\
(2)\ S \to A & : & \begin{array}{l} Text4, E(Kas : TVPa, Kab, B, Text3), \\ E(Kbs : [Ts|Ns], Kab, A, Text2) \end{array} \\
(3)\ A \to B & : & \begin{array}{l} Text6, E(Kbs : [Ts|Ns], Kab, A, Text2), \\ E(Kab : [Ta|Na], B, Text5) \end{array} \\
(4)\ B \to A & : & Text8, E(Kab : [Tb|Nb], A, Text7)
\end{array}
$$

### 6.3.9 ISO Five-Pass Authentication Protocol

$$
\begin{array}{lcl}
(1)\ A \to B & : & Ra, Text1 \\
(2)\ B \to S & : & R'b, Ra, A, Text2 \\
(3)\ S \to B & : & Text5, E(Kbs : R'b, Kab, A, Text4), E(Kas : Ra, Kab, B, Text3) \\
(4)\ B \to A & : & Text7, E(Kas : Ra, Kab, B, Text3), E(Kab : Rb, Ra, Text6) \\
(5)\ A \to B & : & Text9, E(Kab : Ra, Rb, Text8)
\end{array}
$$

### 6.3.10 Woo and Lam Authentication Protocols

The following series of one-way authentication protocols are similar. Some are known to be incorrect. The published accounts of these protocols are given in [117]. Woo and Lam state that a protocol is correct if

> "whenever a responder finishes execution of the protocol, the initiator of the protocol is in fact the principal claimed in the initial message".

Woo and Lam start with a protocol $\Pi_f$ and progressively simplify it to $\Pi$. The final simplification leads to a flawed protocol. Note: in their 1994 paper [117] Woo and Lam state that they assume that principals can detect the replay of messages they have created.

The protocol $\Pi_f$.

(1) $A \to B \; : \; A$

(2) $B \to A \; : \; Nb$

(3) $A \to B \; : \; E(Kas : A, B, Nb)$

(4) $B \to S \; : \; E(Kbs : A, B, Nb, E(Kas : A, B, Nb))$

(5) $S \to B \; : \; E(Kbs : A, B, Nb)$

The protocol $\Pi_1$.

(1) $A \to B \; : \; A$

(2) $B \to A \; : \; Nb$

(3) $A \to B \; : \; E(Kas : A, B, Nb)$

(4) $B \to S \; : \; E(Kbs : A, B, E(Kas : A, B, Nb))$

(5) $S \to B \; : \; E(Kbs : A, B, Nb)$

The protocol $\Pi_2$.

(1) $A \to B \; : \; A$

(2) $B \to A \; : \; Nb$

(3) $A \to B \; : \; E(Kas : A, Nb)$

(4) $B \to S \; : \; E(Kbs : A, E(Kas : A, Nb))$

(5) $S \to B \; : \; E(Kbs : A, Nb)$

The protocol $\Pi_3$.

(1) $A \to B \; : \; A$

(2) $B \to A \; : \; Nb$

(3) $A \to B \; : \; E(Kas : Nb)$

(4) $B \to S \; : \; E(Kbs : A, E(Kas : Nb))$

(5) $S \to B \; : \; E(Kbs : A, Nb)$

The protocol $\Pi$.

(1) $A \to B \; : \; A$

(2) $B \to A \; : \; Nb$

(3) $A \to B \; : \; E(Kas : Nb)$

(4) $B \to S \; : \; E(Kbs : A, E(Kas : Nb))$

(5) $S \to B \; : \; E(Kbs : Nb)$

The protocol $\Pi$ can be attacked as follows:

$$\begin{array}{llcl}
(1) & Z(A) \rightarrow B & : & A \\
(2) & B \rightarrow Z(A) & : & Nb \\
(3) & Z(A) \rightarrow B & : & G \\
(4) & B \rightarrow Z(S) & : & E(Kbs : A, G) \\
(1') & B \rightarrow Z(R) & : & B \\
(2') & Z(R) \rightarrow B & : & Z, E(Kzs : Nb) \\
(3') & B \rightarrow Z(R) & : & E(Kbs : Z, E(Kzs : Nb)) \\
(4') & Z(B) \rightarrow S & : & E(Kbs : Z, E(Kzs : Nb)) \\
(5') & S \rightarrow Z(B) & : & E(Kbs : Nb) \\
(5) & Z(S) \rightarrow B & : & E(Kbs : Nb)
\end{array}$$

Here $Z$ waits for $B$ to start up a protocol run at $(1')$ with some principal $R$ to complete the attack.

Alternatively it may be attacked as follows:

$$\begin{array}{llcl}
(1) & Z(A) \rightarrow B & : & A \\
(1') & Z \rightarrow B & : & Z \\
(2) & B \rightarrow Z(A) & : & Na \\
(2') & B \rightarrow Z & : & Nz \\
(3) & Z(A) \rightarrow B & : & G \\
(3') & Z \rightarrow B & : & E(Kzs : Na) \\
(4) & B \rightarrow S & : & E(Kbs : A, G) \\
(4') & B \rightarrow S & : & E(Kbs : Z, E(Kzs : Na)) \\
(5') & & & \\
(5) & S \rightarrow B & : & E(Kbs : Na)
\end{array}$$

These protocol attacks are indeed given in [117]. However, the protocols would appear to be subject to some straightforward replay attacks. For example, in $\Pi_3$

$$\begin{array}{llcl}
(1) & Z(A) \rightarrow B & : & A \\
(2) & B \rightarrow Z(A) & : & Nb \\
(3) & Z(A) \rightarrow B & : & Nb \\
(4) & B \rightarrow Z(S) & : & E(Kbs : A, Nb) \\
(5) & Z(S) \rightarrow B & : & E(Kbs : A, Nb)
\end{array}$$

Similar attacks may be mounted against $\Pi_1$ and $\Pi_2$ etc. as stated above Woo and Lam assume explicitly that principals can detect replays of messages they have created. Even if this were so (and we would prefer the mechanism to be part of the protocol, a point raised also by Lowe [76]) the security of the protocol would still depend on the properties of the cryptosystem used. Thus, Woo and Lam note that Π is not susceptible to the above form of attack. This is not necessarily the case. If we assume that the symmetric cipher is commutative then we can carry out the following attack:

$$\begin{array}{lrcl}
(\ 1\ ) & B \to Z & : & B \\
(2.1) & Z(A) \to B & : & A \\
(2.2) & B \to Z(A) & : & N \\
(1.2) & Z \to B & : & E(N : Kzs) \\
(1.3) & B \to Z & : & E(E(N : Kzs) : Kbs) \\
(2.5) & Z(S) \to B & : & E(N : Kbs)
\end{array}$$

### 6.3.11 Woo and Lam Mutual Authentication protocol

Here is a protocol due to Woo and Lam [117] that combines mutual authentication and key distribution.

$$\begin{array}{llcl}
(1) & P \to Q & : & P, N1 \\
(2) & Q \to P & : & Q, N2 \\
(3) & P \to Q & : & E(Kps : P, Q, N1, N2) \\
(4) & Q \to S & : & E(Kps : P, Q, N1, N2), E(Kqs : P, Q, N1, N2) \\
(5) & S \to Q & : & E(Kps : Q, N1, N2, Kpq), E(Kqs : P, N1, N2, Kpq) \\
(6) & Q \to P & : & E(Kps : Q, N1, N2, Kpq), E(Kpq : N1, N2) \\
(7) & P \to Q & : & E(Kpq : N2)
\end{array}$$

There is a novel attack on this protocol due to Clark, Jacob and Ryan [37]. Effectively, the principal $Q$ can launch a parallel session attack that causes $P$ to accept as new a previously issued key. The attack consists of the following steps:

$$\begin{array}{rrcl}
(1.1) & P \rightarrow Q & : & P, N1 \\
(2.1) & Q \rightarrow P & : & Q, N1 \\
(2.2) & P \rightarrow Q & : & P, N2 \\
(1.2) & Q \rightarrow P & : & Q, N2 \\
(1.3) & P \rightarrow Q & : & E(Kps : P, Q, N1, N2) \\
(1.4) & Q \rightarrow S & : & E(Kps : P, Q, N1, N2), E(Kqs : P, Q, N1, N2) \\
(1.5) & S \rightarrow Q & : & E(Kps : Q, N1, N2, Kpq), E(Kqs : P, N1, N2, Kpq) \\
(1.6) & Q \rightarrow P & : & E(Kps : Q, N1, N2, Kpq), E(Kpq : N1, N2) \\
(1.7) & P \rightarrow Q & : & E(Kpq : N2) \\
(2.3) & Q \rightarrow P & : & E(Kqs : Q, P, N1, N2) \\
(2.4) & P \rightarrow Q(S) & : & E(Kqs : Q, P, N1, N2), E(Kps : Q, P, N1, N2) \\
(2.5) & Q(S) \rightarrow P & : & E(Kqs : P, N1, N2, Kpq), E(Kps : Q, N1, N2, Kpq) \\
(2.6) & P \rightarrow Q & : & E(Kqs : P, N1, N2, Kpq), E(Kpq : N1, N2) \\
(2.7) & Q \rightarrow P & : & E(Kpq : N2)
\end{array}$$

$Q$ launches a parallel session in response to $P$ initiating the protocol and uses the same nonce in message (2.1). He waits for *P's* reply $N2$ and then uses that as his response in the first protocol. The first protocol then completes as normal. The second protocol proceeds with $Q$ intercepting all communications intended for $S$ and replaying the components of message (1.5) back as those of message (2.5) (with order reversed) to cause $P$ to re-accept the key. This is recorded in [37]. The above is not a particularly strong attack but indicates clearly that the protocol does not provide the authenticity guarantees that it should.

Lowe has recently found a more vicious attack based on the same notion of message component symmetry [76].

## 6.4 Signatures with Conventional Key Encryption

### 6.4.1 Needham-Schroeder Signature Protocol

This protocol comes from the classic paper [87] (where it is *Protocol 3*). Principal $A$ wants to send a message to $B$ and guarantee its origin and integrity. First he forms a characteristic value (digest) $CS$ of the message (using a suitable one-way function). He then sends this to the trusted server $S$ in message (1). $S$, who also holds the key $Kas$ is able to recover the digest $CS$ and form an authenticator (including the *claimed* sender) to be sent to $A$. Note $Kss$ is a key known only to the server $S$. $A$ can then send the message and its authenticator to $B$ who subsequently asks for the authenticator information to be made visible to him. He then calculates the digest of the message received and checks the result against the CS value returned to him by $S$.

$$\begin{array}{lcl}
(1)\ A \rightarrow S & : & A, E(Kas : CS) \\
(2)\ S \rightarrow A & : & E(Kss : A, CS) \\
(3)\ A \rightarrow B & : & Message, E(Kss : A, CS) \\
(4)\ B \rightarrow S & : & B, E(Kss : A, CS) \\
(5)\ S \rightarrow B & : & E(Kbs : A, CS)
\end{array}$$

## 6.5 Symmetric Key Repeated Authentication protocols

### 6.5.1 Kerberos Version 5

This protocol is in three parts each of which is now explained. The protocol involves a user $U$ and four computer principals: a client $C$; a server $S$ with whom $C$ wishes to communicate; and two trusted servers $G$ and $A$. $G$ is known as a *Ticket Granting Server* and provides keys for communication between clients such as $C$ and and servers such as $S$. $A$ is known as the *Key Distribution Centre* and provides keys for communication between clients such as $C$ and ticket granting servers such as $G$. The full protocol has three parts each consisting of two messages between the client $C$ and each of the servers in turn as shown in figure 10. In the protocol descriptions that follow shared secret keys are written with subscripts of the principals who share (or who will share) them. Thus, $Kcg$ denotes the key for secure communication between $C$ and $G$. We use $Ku$ to denote the key used to encrypt communications between $A$ and $C$ on behalf of the user $U$. It is a key obtained by hashing $U's$ password. $A$ stores this password, $C$ will request it from $U$.

The first part of the protocol concerns only $C$ and $A$.

$$\begin{array}{lcl}
(1)\ C \rightarrow A & : & U, G, L1, N1 \\
(2)\ A \rightarrow C & : & U, Tcg, E(Ku : G, Kcg, Tstart, Texpire, N1)
\end{array}$$

where

$$Tcg = E(Kag : U, C, G, Kcg, Tstart, Texpire)$$

In message (1) the client $C$ informs the key distribution centre $A$ that he wishes to communicate on behalf of user $U$ with the ticket granting server $G$. A lifetime $L$ and a nonce $N1$ are sent too. $A$ generates a new key $Kcg$ for this purpose and encrypts it under the key $Ku$ it shares (or will share when $U$ enters his password) with $C$. It also forms a 'ticket'

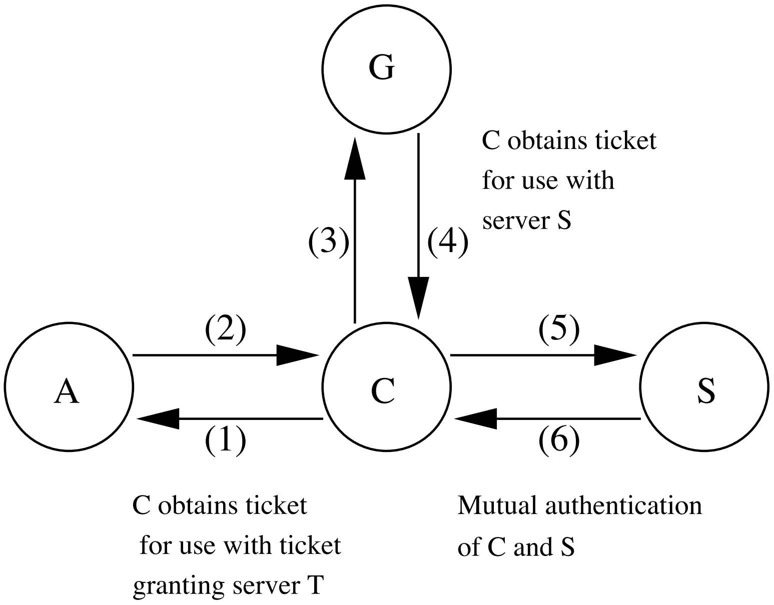


Figure 10: Kerberos Exchanges

*Tcg* that contains the user identity, the client identity, the identity of G, the new key *Kcg* together with timestamp information. The $Tstart, Texpire$ limit the interval over which the ticket is considered as valid. This ticket is encrypted using the key *Kag* shared between *A* and *G*. *C* uses the key *Ku* to decrypt the third component of message (2) and obtains the key *Kcg* which it can now use to communicate with *G*. This is carried out in the second part of the protocol described below:

$$\begin{aligned}(3)\ C \rightarrow G &: S, L2, N2, Tcg, Acg\\ (4)\ G \rightarrow C &: U, Tcs, E(Kcg : S, Kcs, T'start, T'expire, N2)\end{aligned}$$

where

$$Acg = E(Kcg : C, T)$$

$$Tcs = E(Kcg : U, C, S, Kcs, T'start, T'expire)$$

The result of the above is that *G* issues *C* with a ticket *Tcs* and a key *Kcs* to communicate with *S*. The authenticator *Acg* ensures timeliness of message (3).

In the third part of the protocol *C* uses the newly obtained key *Kcs* and ticket *Tcs* to obtain the services of *S*.

$$\begin{aligned}(5)\ C \rightarrow S &: Tcs, Acs\\ (6)\ S \rightarrow C &: E(Kcs : T')\end{aligned}$$

where

$$Acs = E(Kcs : C, T')$$

He forms further authenticator *Acs*, and sends the result to *S* together with the newly acquired encrypted ticket as message (5). *S* carries out decryption on the ticket to obtain the session key *Kcs* and then uses this key to obtain the authentication information. If everything is in order, message (6) is returned.

### 6.5.2 Neuman Stubblebine

This protocol contains two parts: one to bring about the exchange of some ticket and the second is a protocol for multiple authentications. We call these protocols *repeated authentication protocols*. In the Neuman Stubblebine Protocol [90] given below, the first four messages are the initial protocol.

$$\begin{array}{lcl}
(1)\ A \to B & : & A, Na \\
(2)\ \ B \to S & : & B, E(Kbs : A, Na, tb), Nb \\
(3)\ S \to A & : & E(Kas : B, Na, Kab, tb), E(Kbs : A, Kab, tb), Nb \\
(4)\ A \to B & : & E(Kbs : A, Kab, tb), E(Kab : Nb)
\end{array}$$

The repeated authentication part of the protocol is given below.

$$\begin{array}{lcl}
(1)\ A \to B & : & N'a, E(Kbs : A, Kab, tb) \\
(2)\ B \to A & : & N'b, E(Kab : N'a) \\
(3)\ A \to B & : & E(Kab : N'b)
\end{array}$$

Attacks have been successfully mounted on both parts of the protocol. Possible attacks can be found in [60]. The first attack on the initial protocol is given below.

$$\begin{array}{lcl}
(1')\ Z(A) \to B & : & A, Na \\
(2')\ \ B \to Z(S) & : & B, E(Kbs : A, Na, tb), Nb \\
(3') & & \\
(4')\ Z(A) \to B & : & E(Kbs : A, Na(= Kab), tb), E(Na : Nb)
\end{array}$$

The subsequent protocol can then be attacked as follows:

$$\begin{array}{lcl}
(1')\ Z(A) \to B & : & N'a, E(Kbs : A, Na(= Kab), tb) \\
(2')\ B \to Z(A) & : & N'b, E(Kab : N'a) \\
(3')\ Z(A) \to B & : & E(Kab : N'b)
\end{array}$$

The following parallel session attack can be used:

$$\begin{array}{lll}
(1)\ Z(A) \rightarrow B & : & N'a, E(Kbs : A, Kab, tb) \\
(2)\ B \rightarrow Z(A) & : & N'b, E(Kab : N'a) \\
(1')\ Z(A) \rightarrow B & : & N'b, E(Kbs : A, Kab, tb) \\
(2')\ B \rightarrow Z(A) & : & N''b, E(Kab : N'b) \\
(3)\ Z(A) \rightarrow B & : & E(Kab : N'b)
\end{array}$$

In this attack the initial ticket is recorded from a previous legitimate run of the protocol.

### 6.5.3 Kehne Langendorfer Schoenwalder

Here is the KLS repeated authentication protocol. The first five messages form the ticket distribution part. The key *Kbb* is known only to *B*.

$$\begin{array}{lll}
(1)\ A \rightarrow B & : & Na, A \\
(2)\ B \rightarrow S & : & Na, A, Nb, B \\
(3)\ S \rightarrow B & : & E(Kbs : Nb, A, Kab), E(Kas : Na, B, Kab) \\
(4)\ B \rightarrow A & : & E(Kas : Na, B, Kab), E(Kbb : tb, A, Kab), Nc, E(Kab : Na) \\
(5)\ A \rightarrow B & : & E(Kab : Nc)
\end{array}$$

The repeated protocol is:

$$\begin{array}{lll}
(1')\ A \rightarrow B & : & N'a, E(Kbb : tb, A, Kab) \\
(2')\ B \rightarrow A & : & N'b, E(Kab : N'a) \\
(3')\ A \rightarrow B & : & E(Kab : N'b)
\end{array}$$

The repeated authentication part of the protocol is subject to an attack that is identical in form to the parallel session attack on the Neuman Stubblebine repeated protocol.

### 6.5.4 The Kao Chow Repeated Authentication Protocols

In 1995, Kao and Chow proposed a similar repeated authentication protocol that was not susceptible to the attacks on the Neuman Stubblebine protocol [67].

$$\begin{array}{lll}
(1)\ A \rightarrow S & : & A, B, Na \\
(2)\ S \rightarrow B & : & E(Kas : A, B, Na, Kab), E(Kbs : A, B, Na, Kab) \\
(3)\ B \rightarrow A & : & E(Kas : A, B, Na, Kab), E(Kab : Na), Nb \\
(4)\ A \rightarrow B & : & E(Kab : Nb)
\end{array}$$

This protocol suffers when a session key is compromised (as in the Denning Sacco attack on the Needham Schroeder protocol). The authors therefore proposed (in the same paper) to use a different key purely for the handshake. The protocol now becomes:

$$\begin{array}{lcl}
(1)\ A \rightarrow S & : & A, B, Na \\
(2)\ S \rightarrow B & : & E(Kas : A, B, Na, Kab, Kt), E(Kbs : A, B, Na, Kab, Kt) \\
(3)\ B \rightarrow A & : & E(Kas : A, B, Na, Kab), E(Kt : Na, Kab), Nb \\
(4)\ A \rightarrow B & : & E(Kt : Na, Kab)
\end{array}$$

This protocol is further extended to encompass tickets.

$$\begin{array}{lcl}
(1)\ A \rightarrow S & : & A, B, Na \\
(2)\ S \rightarrow B & : & E(Kas : A, B, Na, Kab, Kt), E(Kbs : A, B, Na, Kab, Kt) \\
(3)\ B \rightarrow A & : & \begin{array}{l} E(Kas : A, B, Na, Kab), \\ E(Kt : Na, Kab), Nb, E(Kbs : A, B, Ta, Kab) \end{array} \\
(4)\ A \rightarrow B & : & E(Kt : Na, Kab), E(Kbs : A, B, Ta, Kab)
\end{array}$$

## 6.6 Public Key Protocols without Trusted Third Party

All the ISO protocols in this section may be found in Part 3 of the ISO/IEC 9798 Standard [63].

### 6.6.1 ISO Public Key One-Pass Unilateral Authentication Protocol

$$\begin{array}{lcl}
(1)\ A \rightarrow B & : & Cert_A, [Ta|Na], B, Text2, E(K_a^{-1} : [Ta|Na], B, Text1)
\end{array}$$

### 6.6.2 ISO Public Key Two-Pass Unilateral Authentication Protocol

$$\begin{array}{lcl}
(1)\ B \rightarrow A & : & Rb, Text1 \\
(2)\ A \rightarrow B & : & Cert_A, Ra, Rb, B, Text3, E(K_a^{-1} : Ra, Rb, B, Text2)
\end{array}$$

### 6.6.3 ISO Public Key Two-Pass Mutual Authentication Protocol

$$\begin{array}{lcl}
(1)\ A \rightarrow B & : & Cert_A, [Ta|Na], B, Text2, E(K_a^{-1} : [Ta|Na], B, Text1) \\
(2)\ B \rightarrow A & : & Cert_B, [tb|Nb], A, Text4, E(K_b^{-1} : [tb|Nb], A, Text3)
\end{array}$$

This protocol is in fact two independent applications of the single pass unilateral authentication protocol.

### 6.6.4 ISO Three-Pass Mutual Authentication Protocol

$$\begin{array}{lcl}
(1)\ B \rightarrow A & : & Rb, Text1 \\
(2)\ A \rightarrow B & : & Cert_A, Ra, Rb, B, Text3, E(K_a^{-1} : Ra, Rb, B, Text2) \\
(3)\ A \rightarrow B & : & Cert_B, Rb, Ra, A, Text5, E(K_b^{-1} : Rb, Ra, A, Text4)
\end{array}$$

This is the unilateral two-pass protocol with message (3) added.

### 6.6.5 ISO Two Pass Parallel Mutual Authentication Protocol

$$\begin{array}{lcl}
(1)\ A \rightarrow B & : & Cert_A, Ra, Text1 \\
(1')\ B \rightarrow A & : & Cert_B, Rb, Text2 \\
(2)\ B \rightarrow A & : & Rb, Ra, A, Text6, E(K_b^{-1} : Rb, Ra, A, Text5) \\
(2')\ A \rightarrow B & : & Ra, Rb, B, Text4, E(K_a^{-1} : Ra, Rb, B, Text3)
\end{array}$$

### 6.6.6 Bilateral Key Exchange with Public Key

$$\begin{array}{lcl}
(1)\ B \rightarrow A & : & B, E(K_a : Nb, B) \\
(2)\ A \rightarrow B & : & E(K_b : f(Nb), Na, A, K) \\
(3)\ B \rightarrow A & : & E(K : f(Na))
\end{array}$$

### 6.6.7 Diffie Hellman Exchange

In the Diffie-Hellman algorithm two numbers are publicly agreed by the communicating principals $A$ and $B$. Let these numbers be $G$ and $N$. The protocol is

$$\begin{array}{lcl}
(1)\ A \rightarrow B & : & X = G^x modN \\
(2)\ B \rightarrow A & : & Y = G^y modN
\end{array}$$

$A$ chooses $X = G^x modN$ for some random $x$ and sends the result to $B$ as message (1). $B$ chooses $Y = G^y modN$ for some random $y$ and sends the result to $A$ as message (2). $A$ computes $k = Y^x modN$ and $B$ computes $k = X^y modN$. The result of these two calculations is the same and equal to the new session key. This provides a means of key exchange but no guarantees of authenticity.

## 6.7 Public Key Protocols with Trusted Third Party

### 6.7.1 Needham-Schroeder Public Key Protocol

This protocol appears in the classic paper [87]. It has recently been shown to contain a flaw by Gavin Lowe as part of the project research work.

$$
\begin{array}{lcl}
(1)\ A \rightarrow S & : & A, B \\
(2)\ S \rightarrow A & : & E(K_s^{-1} : K_b, B) \\
(3)\ A \rightarrow B & : & E(K_b : Na, A) \\
(4)\ B \rightarrow S & : & B, A \\
(5)\ S \rightarrow B & : & E(K_s^{-1} : K_a, A) \\
(6)\ B \rightarrow A & : & E(K_a : Na, Nb) \\
(7)\ A \rightarrow B & : & E(K_b : Nb)
\end{array}
$$

Lowe has discovered an attack on this protocol ([74]). Messages 1, 2, 4 and 5 are concerned purely with obtaining public key certificates and are omitted from the description of the attack below:

$$
\begin{array}{lcl}
(3)\quad A \rightarrow Z & : & E(K_z : Na, A) \\
(3')\ Z(A) \rightarrow B & : & E(K_b : Na, A) \\
(6')\ B \rightarrow Z(A) & : & E(K_a : Na, Nb) \\
(6)\quad Z \rightarrow A & : & E(K_a : Na, Nb) \\
(7)\quad A \rightarrow Z & : & E(K_z : Nb) \\
(7')\ Z(A) \rightarrow B & : & E(K_b : Nb)
\end{array}
$$

## 6.8 SPLICE/AS Authentication Protocol

This is a mutual authentication protocol between a client $C$ and a server $S$ using a certification authority $AS$ to distribute public keys where necessary. In the protocol $T$ is a timestamp and $L$ is a lifetime.

$$
\begin{array}{lcl}
(1)\ C \rightarrow AS & : & C, S, N_1 \\
(2)\ AS \rightarrow C & : & AS, E(K_{AS}^{-1} : AS, C, N_1, K_S) \\
(3)\quad C \rightarrow S & : & C, S, E(K_C^{-1} : C, T, L, E(K_S : N_2)) \\
(4)\ S \rightarrow AS & : & S, C, N_3 \\
(5)\ AS \rightarrow S & : & AS, E(K_{AS}^{-1} : AS, S, N_3, K_C) \\
(6)\quad S \rightarrow C & : & S, C, E(K_C : S, N_2 + 1)
\end{array}
$$

This protocol has been shown to be flawed (in different ways) by Hwang and Chen [59] and also Gavin Lowe.

In the first attack it is possible to impersonate a client:

$$\begin{array}{lrcl}
(1) & Z \rightarrow AS & : & Z, S, N_1 \\
(2) & AS \rightarrow Z & : & AS, E(K_{AS}^{-1} : AS, Z, N_1, K_S) \\
(3) & Z(C) \rightarrow S & : & C, S, E(K_Z^{-1} : C, T, L, E(K_S : N_2)) \\
(4) & S \rightarrow Z(AS) & : & S, C, N_3 \\
(4') & Z(S) \rightarrow AS & : & S, Z, N_3 \\
(5) & AS \rightarrow S & : & AS, E(K_{AS}^{-1} : AS, S, N_3, K_Z) \\
(6) & S \rightarrow Z(C) & : & S, C, E(K_Z : S, N_2 + 1)
\end{array}$$

In the second attack it is possible to impersonate the server:

$$\begin{array}{lrcl}
(1) & C \rightarrow Z(AS) & : & C, S, N_1 \\
(1') & Z(C) \rightarrow AS & : & C, Z, N_1 \\
(2) & AS \rightarrow C & : & AS, E(K_{AS}^{-1} : AS, C, N_1, K_Z) \\
(3) & C \rightarrow Z(S) & : & C, S, E(K_C^{-1} : C, T, L, E(K_Z : N_2)) \\
(4) & Z \rightarrow AS & : & Z, C, N_3 \\
(5) & AS \rightarrow Z & : & AS, E(K_{AS}^{-1} : AS, Z, N_3, K_C) \\
(6) & Z(S) \rightarrow C & : & S, C, E(K_C : S, N_2 + 1)
\end{array}$$

In the third attack (by Gavin Lowe) message (3) is replayed within the possible time window to re-achieve authentication.

### 6.8.1 Hwang and Chen's Modified SPLICE/AS

Hwang and Chen [59] proposed an enhanced protocol to overcome the flaws (that they had identified) in SPLICE protocol presented above. This modified SPLICE/AS protocol has recently been shown by Clark and Jacob to be flawed too.

$$\begin{array}{lrcl}
(1) & C \rightarrow AS & : & C, S, N_1 \\
(2) & AS \rightarrow C & : & AS, E(K_{AS}^{-1} : AS, C, N_1, S, K_S) \\
(3) & C \rightarrow S & : & C, S, E(K_C^{-1} : C, T, L, E(K_S : N_2)) \\
(4) & S \rightarrow AS & : & S, C, N_3 \\
(5) & AS \rightarrow S & : & AS, E(K_{AS}^{-1} : AS, S, N_3, C, K_C) \\
(6) & S \rightarrow C & : & S, C, E(K_C : S, N_2 + 1)
\end{array}$$

For the purposes of the attack we need only consider messages (3) and (6) and so we assume that all public keys are appropriately obtained or possessed.

$$\begin{array}{lrcl}
(3) & C \rightarrow Z(S) & : & C, S, E(K_C^{-1} : C, T, L, E(K_S : N_2)) \\
(3') & Z \rightarrow S & : & Z, S, E(K_Z^{-1} : Z, T, L, E(K_S : N_2)) \\
(6') & S \rightarrow Z & : & S, Z, E(K_Z : S, N_2 + 1) \\
(6) & Z(S) \rightarrow C & : & S, C, E(K_C : S, N_2 + 1)
\end{array}$$

The problem arises because the server $S$ is fooled as to the origin of the encrypted nonce in (3′). It is created by $C$ in (3) but is used by $Z$ in (3′) who pretends to have created it himself.

## 6.9 Denning Sacco Key Distribution with Public Key

This protocol is intended to provide a means for a conventional communications key $Kab$ to be generated by $A$ and passed to $B$ securely. $S$ provides certificates to $A$ and $B$.

$$\begin{aligned}
&(1)\ A \rightarrow S &:&\quad A, B\\
&(2)\ S \rightarrow A &:&\quad Cert_A, Cert_B\\
&(3)\ A \rightarrow B &:&\quad Cert_A, Cert_B, E(K_b : E(K_a^{-1} : Kab, Ta))
\end{aligned}$$

where $Cert_A = E(K_s^{-1} : A, K_a, T)$ is the public key certificate of $A$ signed by $S$ etc. There is a problem with this protocol (discovered by Abadi in 1994). $B$ can now decrypt to obtain the session key and timestamp signed by $A$ and form a message of the form

$$(3)\ B(A) \rightarrow C \quad : \quad Cert_A, Cert_C, E(K_c : E(K_a^{-1} : Kab))$$

and can now masquerade as $A$ to $C$.

### 6.9.1 CCITT X.509

This is the classic description, as it appears in [26], of three protocols (consisting either of message 1, messages 1 and 2 or all three below. It has been shown to be flawed.

$$\begin{aligned}
&(1)\ A \rightarrow B &:&\quad A, E(K_a^{-1} : Ta, Na, B, X_a, E(K_b : Y_a))\\
&(2)\ B \rightarrow A &:&\quad B, E(K_b^{-1} : tb, Nb, A, Na, X_b, E(K_a : Y_b))\\
&(3)\ A \rightarrow B &:&\quad A, E(K_a^{-1} : Nb)
\end{aligned}$$

Attacks have been found by L'Anson and Mitchell [12] and by the Burrows Abadi and Needham [26]. The problem is that there is signing after encryption. If an encrypted message has a component that is itself encrypted under a public key then it cannot be deduced that the sender actually knows the contents of that component.

## 6.10 Miscellaneous

### 6.10.1 Shamir Rivest Adelman Three Pass protocol

The following protocol differs in that the participants share no secrets. It was suggested as a means of transmitting data over an insecure channel. It assumes that encryption is commutative. It is known to be subject to a variety of attacks.

$$\begin{array}{lcl}
(1)\ A \rightarrow B & : & E(K_a : M) \\
(2)\ B \rightarrow A & : & E(K_b : E(K_a : M)) \\
(3)\ A \rightarrow B & : & E(K_b : M)
\end{array}$$

The first attack simply uses $A$ as an oracle.

$$\begin{array}{lcl}
(1)\ A \rightarrow Z(B) & : & E(K_a : M) \\
(2)\ Z(B) \rightarrow A & : & E(K_a : M) \\
(3)\ A \rightarrow Z(B) & : & M
\end{array}$$

Carlsen suggests that it might be possible to simply check whether the message returned in (2) is in fact encrypted, but there would seem to be a very simple attack, namely one where a legitimate principal $C$ takes on the role of $B$ but using his own key.

There is also another attack:

$$\begin{array}{lcl}
(1)\ A \rightarrow Z(B) & : & E(K_a : M) \\
(1')\ Z(B) \rightarrow A & : & E(K_a : M) \\
(2')\ A \rightarrow Z(B) & : & M \\
(2)\ Z(B) \rightarrow A & : & bogus \\
(3)\ A \rightarrow Z(B) & : & E(K_a : bogus)
\end{array}$$

### 6.10.2 Gong Mutual Authentication Protocol

This protocol [50] is based on the use of one-way functions rather than encryption. In the following protocol $f$ and $g$ are both one-way (publicly known) functions (they may be identical). Each principal $A$ and $B$ shares a secret, $P_a$ and $P_b$ respectively, with the authentication server $S$. $Na$, $Nb$ and $N_s$ are nonces.

$$\begin{array}{lcl}
(1)\ A \rightarrow B & : & A, B, Na \\
(2)\ B \rightarrow S & : & A, B, Na, Nb \\
(3)\ S \rightarrow B & : & N_s, f(N_s, Nb, A, P_b) \oplus (k, h_a, h_b), g(k, h_a, h_b, P_b) \\
(4)\ B \rightarrow A & : & N_s, h_b \\
(5)\ A \rightarrow B & : & h_a
\end{array}$$

In message (3) $(k, h_a, h_b) = f(N_s, Na, B, P_a)$ is calculated by the server $S$. $k$ is a secret to be shared between $A$ and $B$, while $h_a$ and $h_b$ are called *handshake* numbers. The symbol $\oplus$ represents the XOR function. Principal $B$ computes $f(N_s, Nb, A, P_b)$ to retrieve $(k, h_a, h_b)$ from the second item of the message. It also computes $g(k, h_a, h_b, P_b)$ to check against the third item that tampering has not taken place. After receiving message (4). $A$ computes $f(N_s, Na, B, P_a)$ to get $(k, h_a, h_b)$ . If the value of $h_b$ matches the one sent by $B$ then $A$ replies with message (5). The literature surveyed has not indicated that the protocol is flawed.

### 6.10.3 Encrypted Key Exchange – EKE

This is an unusual protocol due to Bellovin and Merritt [15] and has the following steps:

$$\begin{array}{lcl}
(1)\ A \rightarrow B & : & E(P : K_a) \\
(2)\ B \rightarrow A & : & E(P : E(K_a : R)) \\
(3)\ A \rightarrow B & : & E(R : Na) \\
(4)\ B \rightarrow A & : & E(R : Na, Nb) \\
(5)\ A \rightarrow B & : & E(R : Nb)
\end{array}$$

Here $P$ is a password used as a symmetric key, $K_a$ is a randomly generated public key. $R$ is a randomly generated session key. There would appear to be a fairly straightforward parallel session attack on the above protocol (unreported in the literature)

$$\begin{array}{lcl}
(1.1)\ A \rightarrow Z(B) & : & E(P : K_a) \\
(2.1)\ Z(B) \rightarrow A & : & E(P : K_a) \\
(2.2)\ A \rightarrow Z(B) & : & E(P : E(K_a : R)) \\
(1.2)\ Z(B) \rightarrow A & : & E(P : E(K_a : R)) \\
(1.3)\ A \rightarrow Z(B) & : & E(R : Na) \\
(2.3)\ Z(B) \rightarrow A & : & E(R : Na) \\
(2.4)\ A \rightarrow Z(B) & : & E(R : Na, Nb) \\
(1.4)\ Z(B) \rightarrow A & : & E(R : Na, Nb) \\
(1.5)\qquad A \rightarrow B & : & E(R : Nb) \\
(2.5)\ Z(B) \rightarrow A & : & E(R : Nb)
\end{array}$$

### 6.10.4 Davis Swick Private Key Certificates

The first protocol given by Davis and Swick [40] is for key translation via a trusted translator $T$. The protocol is given below:

$$\begin{aligned}
&(1)\ B \rightarrow A &:&\quad E(Kbt : A, msg)\\
&(2)\ A \rightarrow T &:&\quad E(Kbt : A, msg), B\\
&(3)\ T \rightarrow A &:&\quad E(Kat : msg, B)
\end{aligned}$$

On receiving message (3) $A$ assumes that $msg$ originated with $B$ and was destined for $A$. If $B$ arranges for $msg = CX$ for some identifier $C$ then message (3) becomes $E(C, XB : Kat)$. $B$ can now use this to masquerade as $A$ in message (1). Sufficient redundancy would need to be placed in the message to prevent this attack (noticed by Clark and Jacob). The protocol does not ensure timeliness.

A scaled up version of the key translation service is also presented.

$$\begin{aligned}
&(1)\ B \rightarrow A &:&\quad E(Kbt : A, msg)\\
&(2)\ A \rightarrow T &:&\quad E(Kbt : A, msg), E(Kt : Kbt, B, Lb), E(Kt : Kat, A, La)\\
&(3)\ T \rightarrow A &:&\quad E(Kat : msg, B)
\end{aligned}$$

Here $Kt$ is $T's$ master key used to signed the key containing tickets. $La$ and $Lb$ are lifetimes. $E(Kat : Kat, A, La)$ is *$A's$ private key certificate* created by $T$.

There is a key distribution protocol:

$$\begin{aligned}
&(1)\ B \rightarrow T &:&\quad E(Kt : Kbt, B, Lb)\\
&(2)\ T \rightarrow B &:&\quad E(kt : K'bt, B, L'b), E(Kbt : K'bt, T, L'b, checksum)
\end{aligned}$$

Another key distribution protocol is given:

$$\begin{aligned}
&(1)\ A \rightarrow T &:&\quad E(Kt : Kat, A, La), encKbt, B, LbKt\\
&(2)\ T \rightarrow A &:&\quad E(Kbt : Kab, A, Lab)E(Kat : Kab, B, Lab, checksum)\\
&(3)\ A \rightarrow B &:&\quad E(Kab : msg, A), E(Kbt : Kab, A, Lab)
\end{aligned}$$

# 7 Acknowledgements

This survey was carried out as part of a Strategic Research Plan managed by Peter Ryan of the Defence Evaluation Research Agency. The authors would like to express their thanks for supporting this work. In addition, we would like to thank Peter Ryan, Irfan Zakuiddin, Gavin Lowe and Colin Runciman for their helpful comments. John Clark is Lecturer in Critical Systems and Jeremy Jacob is Lecturer in Computer Science at the University of York.

# References

[1] Martin Abadi and Roger Needham. Prudent Engineering Practice for Cryptographic Protocols. Technical report, SRC DIGITAL, June 1994.

This paper sets out several heuristic rules which enable people to write good protocols, though the report says that they are neither sufficient or necessary. Rather, extant practice has shown their utility.

The paper is an excellent piece of work and will prove of considerable use to protocol designers. The paper does not go into detail as to what the "goals" of a protocol are or should be, how this should be stated or verified. It is an engineering account of protocol development whose principles, if applied judiciously, will lead to better protocols. The guidelines serve as a "have you remembered this sort of attack" list for the designer.

There are two overarching principles.

**Principle 1** Every message should say what it means; its interpretation should depend only on its content.

**Principle 2** The conditions for a message to be acted upon should be clearly set out so that someone reviewing a design may see whether they are acceptable or not.

The first principle rules out different interpretations of received messages due to (for example) different assumptions of message formats. Principle 2 seems clear, and helps the designers of the individual protocol engines.

The remaining principles are:

**Principle 3** If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name in the message.

Two examples are given (the attack on the Denning and Sacco Asymmetric Key Exchange protocol and the Woo and Lam protocol for checking the existence of a principle).

**Principle 4** Be clear about why encryption is being done. Encryption is not wholly cheap and not asking precisely why it is being done can lead to redundancy. Encryption is not synonymous with security and its improper use can lead to errors.

The point is illustrated with reference to a simplified form of the Kerberos protocol.

**Principle 5** When a principal signs material that has already been encrypted it should not be inferred that the principal knows the content

of he message. On the other hand, it is proper to infer that the principal that signs a message and then encrypts it for privacy knows the content of that message.

Failures arising from ignoring this principle are given by reference to the CCITT.509 one message protocol.

**Principle 6** Be clear what you are assuming about nonces. What may do for avoiding temporal succession may not do for ensuring association, and perhaps association is best established by other means.

**Principle 7** The use of a predictable quantity such as the value of a counter can serve in guaranteeing newness, through a challenge response exchange. But if a predictable quality is to be effective it should be protected so that an intruder cannot simulate a challenge and later replay a response.

**Principle 8** If timestamps are used as freshness guarantees by reference to absolute time then the difference between local clocks at various machines must be much less than the allowable age of a message deemed to be valid. Furthermore the time maintenance mechanism everywhere becomes part of the trusted computing base.

**Principle 9** A key may have been used recently, for example to encrypt a nonce, yet be quite old, and possibly compromised. Recent use does not make the key look any better than it would otherwise.

**Principle 10** If an encoding is used to present the meaning of a message, then it should be possible to tell which encoding is being used. In the common case where the encoding is protocol dependent, it should be possible to deduce that the message belongs to this protocol, and in fact to a particular run of the protocol, and to know its number in the protocol.

[2] J. Adam. Cryptography = Privacy? *IEEE Spectrum*, pages 29–35, August 1992.

A racy bit of journalism providing an account of the debate surrounding the Digital Signature Standard. The views of the NSA are recorded regarding its role in the promotion of DSS (and the dropping of RSA) together with a rebuttal by Ron Rivest. A good background read.

[3] Selim G. Akl. Digital Signatures: A Tutorial Survey. *Computer*, pages 15–24, February 1983.

This paper provides an overview of various forms of digital signatures and their implementation (in 1983). Public, private and hybrid approaches to digital signatures are described. Both true and and arbitrated schemes are given. A good overview.

[4] Ross Anderson. UEPS – A Second Generation Electronic Wallet. In Y. Deswarte, G. Eizenberg, and J.-J. Quisquater, editors, *Proceedings ESORICS 92*. Springer LNCS 648, 1992.

In this early paper, the author describes an electronic smartcard application for the banking industry – the Universal Electronic Payments System (UEPS). The approach uses key chaining. An outline is given of attempts to apply BAN logic to the analysis of UEPS is described.

[5] Ross Anderson. Why Cryptosystems Fail. *Comunications of the ACM*, November 1994.

This paper provides a worrying account of how ATM failures have occurred in practice. The author explains a variety of external and internal attacks on ATMs. The author argues very strongly that a *systems* view needs to be taken and that most current stakeholders do not promote this. The goals of ATM systems need to be reconsidered; present approaches have been influenced adversely by military security concepts. Human factors, controlling internal and external fraud and providing effective arbitration means are important. Much of current research is spent on topics pretty much irrelevant to real concerns. Furthermore, it would appear that in practice the military sector is prone to systems failures. Anderson maintains that the TCSEC and ITSEC approaches are currently inappropriate for real-world needs.

[6] Ross Anderson. Making Smartcard Systems Robust. In *Cardis 94*, 1994.

The author outlines aspects that need to be taken into account when designing cryptographic protocols. After providing a categorisation of the areas where attacks can occur the author returns to a familiar theme (that of robustness). The Universal Electronic Payment System (UEPS) is proposed as an example of robust design.

[7] Ross Anderson. Liability and Computer Security: Nine Principles. In Dieter Gollman, editor, *Proceedings ESORICS 94*, pages 231–245. Springer Verlag LNCS 875, 1994.

The author talks about the liability aspect of computer security systems. Several ’non-technical’ aspects of security are addressed such as how computer evidence will stand up in court, legal differences between the USA and the UK. Nine principles are given to guide the process of creating secure systems.

[8] Ross Anderson. Crypto in Europe – Markets, Policy and Law. A paper from RA's WWW Page — http://www.cl.cam.ac.uk/users/rja14/.

This presents an overview of policy on matters cryptographic in various European states. It indicates how the cryptographic debate has become hooked on confidentiality. Authenticity and integrity are much more important problems. There's a good list of real-word applications of cryptography (ranging from ATMs and utility tokens to lottery ticketing terminals and postal franking machines). There are quite a few juicy examples of dire failure of cryptosystems (of which the author has a very large collection). The paper ends with an attack on the currently expressed views about Government restrictions on cryptography. The author argues that villains will in general not use cryptography, that the use of wiretaps varies enormously between between countries, that maintaining wiretaps in digital exchanges is very expensive and that the real threat to the individual arises from unauthorised access to data and has little to do with listening in on encrypted communications.

[9] Ross Anderson and Johan Bezuidenhout. On the Reliability of Electronic Payment Systems. A paper from RA's WWW Page — http://www.cl.cam.ac.uk/users/rja14/, 1995.

In this paper the authors describe attempts to introduce pre-payment electricity meters in South Africa. The paper provides good background information and highlights some novel attacks on meters.

[10] Ross Anderson and Roger Needham. Programming Satan's Computer. A paper from RA's WWW Page — http://www.cl.cam.ac.uk/users/rja14/, 1995.

This paper provides an interesting introduction to how security protocols can fail. The paper spans simple attacks on cash cards to some interesting attacks on protocols themselves. The examples are informative and up-to-date (including various unfortunate features of the Woo and Lam protocols [116], Lowe's attack on the Needham Schroeder public key protocol [74], Abadi's attack on the Denning Sacco public key protocol, the ding-dong attack on the Wide Mouthed Frog protocol (discovered independently by several researchers) and other attacks.

A good engineering approach to protocol development is indicated complementing formal appraoches. Rules of thumb are given for development. An overarching principle of good design is proffered: the *Explicitness Principle*. A good practical read.

[11] Ross Anderson and Roger Needham. Robustness Principles for Public Key Protocols. A paper from RA's WWW Page — http://www.cl.cam.ac.uk/users/rja14/, 1995.

This paper provides many principles or rules of thumb that should be used when developing cryptographic protocols. The principles are generally motivated by some good wholesome attacks. There is a general Principle of Explicitness (that one must be explicit about any properties that may be used to attack a public key primitive as well as typing freshness and any other assumptions being made).

[12] Colin l'Anson and Chris Mitchell. Security Defects in the CCITT Recommendation X.509 — The Directory Authentication Framework. *Computer Communication Review*, 20(2):30–34, April 1990.

This paper presents several problems in the proposed CCITT X509 standard and offers some solutions. The problems indicated are:

- that the particular requirements of clause 8.1 virtually mandate the use of RSA (since it requires that both keys in a pair be usable for encipherment);
- there is a problem arising because the first message encrypts before signing which allows an intruder to simply record encrypted data (he does not know the contents) and include it in a message of his own (the encrypted data encrypted using the recipient's public key);
- the three-way protocol allows an attack that is a mixture of a parallel session attack and a classic replay (this arises because the standard states that the use of timestamps is optional);
- the wording of the conditions for using RSA are incorrect (or rather the rationale is wrong) and a recommended hash function has been shown to be defective.

A good read. Furthermore, the signing after encryption problem has reared its head in other places (for example, Clark and Jacob have recently shown [34] that the SPLICE/AS protocol and an improved version of it, both described in [59], have this flaw.

[13] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of Cipher Block Chaining. In Yvo G. Desmedt, editor, *Advances in Cryptology - Crypto 94*, number 839 in LNCS, pages 341–358. Springer Verlag, 1994.

This paper assesses the usefulness of using CBC approaches for message authentication codes. One of the few papers on the subject. A good read. See also [106].

[14] S. M. Bellovin and M. Merritt. Limitations of the Kerberos Authentication System. *Computer Communication Review*, 20(5):119–132, October 1990.

In this paper the authors describe several weaknesses of the then current Kerberos authentication system. The authors address potential replay attacks, secure time service provision, password guessing attacks, login spoofing, chosen plaintext attacks *inter alia*. Solutions are suggested and assessed. Some interesting remarks are made concerning susceptibility to ciphertext splicing type attacks. A very good read.

[15] S. M. Bellovin and M. Merritt. Encrypted Key Exchange: Password based protocols secure against dictionary attacks. In *Proceedings 1992 IEEE Symposium on Research in Security and Privacy*, pages 72–84. IEEE Computer Society, May 1992.

This paper introduces the encrypted key exchange protocol (EKE). It is unusual in that it uses passwords (a weak secret) to distribute random 'public' keys. A session key is generated and encrypted under the public key and then the password. A challenge response protocol is carried out using the session key. The idea is to make the protocol robust against brute force attacks. It should not be possible to verify a guess for a password etc. There are some technical difficulties involved (e.g. the leakage of information that arises because of primality of public keys etc.). The protocol is discussed with an assessment of its resistance to attacks for RSA and El Gamal.

[16] P. Bieber. A Logic of Communication in a Hostile Environment. In *Proceedings of the Computer Security Foundations Workshop III*, pages 14–22. IEEE Computer Society Press, 1990.

This paper provides a quantified extension to the logic KT5 which also has communication operators (for expressing the sending and receiving of messages). Examples of how to express secrecy and authentication properties are given.

[17] Pierre Bieber and Nora Boulahia-Cuppens. Formal development of authentication protocols. 1993.

This represents the first attempt to apply the B method to the specification and design of authentication protocols. A communications model is provided and proofs are discharged. The B method looks promising.

[18] R. Bird, I. Gopal, A. Herzberg, P. A. Janson, S. Kutten, R. Mulva, and M. Yung. Systematic Design of Two-Party Authentication Protocols. In J. Feigenbaum, editor, *Proceedings of Crypto '91: Advances*

*in Cryptology*, number 576 in Lecture Notes in Computer Science, pages 44–61. Springer Verlag, 1991.

This paper provides a loose heuristic method for searching for flaws in cryptographic protocols. It describes the derivation and analysis of a two-party protocol that is claimed to be as secure as the cipher block chaining encryption that supports it. The protocol looks secure and compact.

[19] R. Bird, I. Gopal, A. Herzberg, P. A. Janson, S. Kutten, R. Mulva, and M. Yung. Systematic Design of a Family of Attack-Resistant Authentication protocols. *IEEE Journal on Selected Areas in Communications*, 11(5):679–693, 1993.

The paper provides a brief introduction to attacks on authentication protocols and sets about developing a 3-pass mutual authentication protocol that avoids the attacks identified. The approach catches many of the principles which have now become prudent practice. It is an informal development.

[20] A. D. Birrell. Secure Communication using Remote Procedure Calls. *ACM Transactions on Operating Systems*, 3(1):1–14, February 1985.

The original description of the Andrew Secure RPC Protocol. An analysis of this protocol can be found in [26].

[21] Colin Boyd. Hidden Assumptions in Cryptographic Protocols. *Proceedings of the IEE*, 137 Pt E(6):433–436, November 1990.

A good, simple and informative paper. The theme is that cryptographic protocol specifications do not state precisely the assumptions they make about the underlying cryptoalgorithms. Two very well known protocols are given (Needham-Schroeder [88] and Otway-Rees [91]) and are shown to depend crucially on the underlying cryptoalgorithm. The dangers of using Cipher Block Chaining (CBC) mode for DES (and others) is shown to allow various attacks (which depend on the implementation). An entertaining and very simple example is given of how use of a stream cipher for the use of challenge response can lead to an attack (both challenge and response are encrypted using the same key). The response is one less than the value of the nonce challenge and the use of a stream cipher means that simply changing the final bit in the encrypted challenge has a half chance of producing the required encrypted response. The author provides new descriptions of what properties are required of cryptoalgorithms and conclude that nonces should not be encrypted in more than one message using the same key.

Problems with CBC mode of encryption are in fact worse than this paper indicates. The paper is well-written and highlights a much neglected area, namely that of the assumptions that protocol designers (implicitly) make about implementations.

[22] Colin Boyd. A Class of Flexible and Efficient Key Management Protocols. In *Proceedings 9th IEEE Computer Security Foundations Workshop*, pages 2–8. IEEE Computer Society Press, 1996.

This paper gives a novel scheme for the distribution of session keys between communicating principals. A trusted server $S$ is used to distribute a key for use in a cryptographic hashing function. The principals exchange nonces which are concatenated and then hashed. The resulting hashed value is the session key. Key compromise is an issue and Boyd suggests ways in which this can be handled. The scheme can be used to create very efficient protocols.

[23] Colin Boyd and Wenbo Mao. On a Limitation of BAN Logic. In Tor Helleseth, editor, *Eurocrypt '93*, number 765 in LNCS, pages 240–247. Springer Verlag, 1993.

Boyd and Mao identify some more limitations of the BAN logic type approach to analysis. The paper presents two examples of 'flawed' protocols. The first is of interest in that it presents a plausible (but faulty) implementation of the intent of the protocol. The authors conclude that post-hoc analysis is not the thing to do; rather correctness by design should be the aim. The criticisms made in this paper were countered at the rump session at the same conference by van Oorschot [120].

[24] Colin Boyd and Wenbo Mao. Designing Secure Key Exchange Protocols. In Dieter Gollmann, editor, *Computer Security—ESORICS '94*, pages 93–106. Springer-Verlag, 1994.

This paper takes the view that program derivation techniques ought to be applied to security protocol derivation, i.e. start with an abstract model and refine it. It introduces a notation for passing restricted sets of messages between principals. A formal model is outlined in the formal specification notation Z. The model aims to provide a moderately general notion of what the effects of sending and receiving messages should be. In brief, there is an abstract model of all $(key, principal)$ pairs generated by trusted principals and each principal has his local view of such pairs known to him. When a new key is generated for a set of principals $U$ then all pairs of the form $(k, u)$, with $u \in U$, are added to the global store. The server may send a message containing the key (together with an indication of the set $U$) to any user in $U$.

There are only two forms of concrete exchange messages and the paper shows how these can be used to model different types of exchange (user-user, protocols using a trusted party and conference key protocols).

The paper identifies issues that are not dealt with (such as key revocation, the effect of untrustworthy behaviour of a principal and more complex distribution protocols).

The formal model as it currently stands needs some adjustment. The problem lies with their definition of security for states of the system. Whilst it would appear true that given a suitable initial state and the definitions of the send and receive operations the resulting operation will be "secure"—the notion of suitable definition of initial state must be addressed. It is perfectly feasible to set up an initial state (and in general such a state will not be a pathological one, i.e. it will have some keys) that is useless. For example, a state where the only pair in the whole system is $(k1, \mathit{Charles})$ and this pair is in the local store of both *Alice* and *Bob*. It is a relatively trivial matter to alter the definition of the security criterion to rule out this sort of possibility though. A good paper and one of the few to talk about deriving protocols rather than post-hoc verifying them.

[25] E. F. Brickell and A. M. Odlyzko. Cryptanalysis: A Survey of Recent Results. *Proceedings of the IEEE*, 76(5), May 1988.

This paper provides an excellent overview of some advanced (in 1988) attacks on a variety of algorithms. A number of attacks are described on knapsack variants, Ong-Schnorr-Shamir and Okamaoto-Shiraishi signatures schemes, RSA and others. It also addresses the Data Encryption Standard.

[26] Michael Burrows, Martin Abadi, and Roger Needham. A Logic of Authentication. Technical Report 39, Digital Systems Research Center, February 1989.

We give a section by section account of this paper. This may seem a little excessive but the paper is clearly the most important paper in the field.

**Section 1** Authentication protocols guarantee that if principals really are who they say they are then they will end up in possession of one or more shared secrets, or at least be able to recognise the use of others' secrets.

There are lots of authentication protocols. It is not clear precisely what these protocols achieve. As a result a formal approach is needed to explain precisely what assumptions are being made within a protocol and what conclusions can legitimately be derived from the successful execution of the protocol.

Some aspects of authentication protocols have been deliberately ignored (no attempt to cater for authentication of untrustworthy principals and no analysis of encryption scheme strength).

The authors are fairly limited in what they claim for the logic that follows:

> *Our goal, however, is not to provide a logic that would explain every authentication method, but rather a logic that would explain most of the central concepts in authentication.*

This is important as BAN logic has often been unfairly criticised.

The authors then give informal accounts of some important notions in authentication

- If you've sent Joe a number that you have never used for this purpose before and if you subsequently receive from Joe something that depends on knowing that number then you ought to believe that Joe's message originated recently — in fact, after yours.
- If you believe that only you and Joe know $K$ then you ought to believe that anything you receive encrypted with $K$ as key comes originally from Joe.
- If you believe that $K$is Joe's public key, then you should believe that any message you can decrypt with $K$ comes originally from Joe.
- If you believe that only you and Joe know $X$ then you ought to believe that any encrypted message that you receive containing $X$ comes originally from Joe.

**Section 2** In this section the authors present their formalism based on a many-sorted modal logic. Messages are regarded as statements in the logic. There are principals, keys and formulae. A number of logical constructs are given.

The authors assume explicitly that a principal is able to detect and ignore message he has sent. The logic is monotonic within a protocol run (that is, beliefs that hold at the start of the run hold at the end). Moreover, the logic assumes that if a principal utters a formula $X$ then he actually believes it.

The authors state "each encrypted message contains sufficient redundancy to be recognised and decrypted unambiguously". This idea of recognisability will be taken up by other authors. Indeed, the notion is a subtle an important one. The notational convenience of omitting the sender of a message is often used. It is, of course the case, that decryption will be necessary if the actual sender of a message is to be known: that is: the message must have sufficient authenticity.

The authors then present a set of postulates fairly modestly: "we do not present the postulates in the most general form possible; our main concern is to have enough machinery to carry out some realistic examples and to explain the method".

Some of the constructs introduced by the authors in this section have a notion of trust involved. The nature of this trust is not made explicit. From the examples, however, it can be seen that trusting a principal to know a shared secret means that he will not release it himself; if this were not the case then many of the later postulates do not make sense. The formalism assumes that all sessions with a shared key are between two parties $P$ and $Q$. Multiple party sessions are of course a practical possibility.

In the description of the nonce verification rule (if I believe that $X$ is fresh and that you have uttered $X$, then I should believe that you believe $X$, because you must have uttered $X$ recently and hence still believe it) the authors suggest that they could introduce a "recently said" operator to overcome the restriction that $X$ must be cleartext. This idea will be taken up by other authors.

The authors then present the notion of an idealised protocol. Standard description of protocols give a fairly concrete description of what bits go where in a message. This is not particularly useful for logical manipulation and so the authors transform each protocol message into a formula. Parts of the formula which do not contribute to the beliefs are omitted; thus there is no cleartext in BAN messages. Each protocol is a sequence of encrypted formulæ. The authors claim that their idealised formulæare clearer and more complete than other traditional descriptions. They also state that deriving an encoding from an idealised protocol is far less time consuming and error prone than understanding the meaning of a particular informal encoding. Omitting cleartext gives rise to some problems, e.g. the direct leakage of information.

Loosely speaking, a message $m$ can be interpreted as a formula $X$ if whenever the recipient gets $m$ he may deduce that the sender must have believed $X$ when he sent the message. This process is fairly controversial. There would appear to be an implicit assumption that we choose the strongest feasible formula for $X$. Failure to do this may require the addition of initial assumptions that would not be necessary under an alternative idealisation. It seems that in addition to iteration of initial beliefs for the purposes of proof, as suggested by the authors, one might well iterate over idealisation too.

The protocol analysis takes the following steps

1. The idealised protocol is derived form the original one.
2. Assumptions about the initial state are written.
3. Logical formulae are attached to statements of the protocol, as assertions about the state of the system after each statement.

4. The logical postulates are applied to the assumptions and the assertions, in order to discover beliefs held by the parties in the protocol.

Effectively we produce an annotated protocol in much the same way as we could produce an annotated program.

The authors state that there is no (refined) notion of time in their logic, nor do they address concurrency issues.

**Section 3** The authors state that there is room for debate as to what the goals of an authentication protocol should be. Several plausible candidates are suggested, the actual goals will of course be system specific.

**Sections 4–11** These sections apply the BAN logic presented to several protocols:

**Section 4** The Otway-Rees Protocol

**Section 5** The Needham Schroeder Protocol with conventional keys.

**Section 6** The Kerberos Protocol

**Section 7** The Wide-mouthed Frog Protocol

**Section 8** The Andrew Secure RPC Handshake

**Section 9** The Yahalom Protocol

**Section 10** The Needham-Schroeder Public Key Protocol

**Section 11** The CCITT X.509 Protocol(s)

We shall not describe the analyses in detail here.

**Sections 12–13** The remaining sections show how the logic can be extended to handle hashing and provide a more formal semantics for the logic.

This paper is essential reading. Most security protocol papers reference it, or one of its other forms, and several criticise it (some more fairly than others). The paper is well written and provides the basis for numerous extensions.

[27] Michael Burrows, Martin Abadi, and Roger M. Needham. Rejoinder to Nessett. *ACM Operating Systems Review*, 24(2):39–40, April 1990.

This is the BAN authors' refutation of Nessett's criticism [89]. They quote from their paper [26] that they did not intend to deal with security issues such as "unauthorised release of secrets". This would appear justified. The

authors state that Nessett's example "accurately points out an intended limitation or our logic" but indicate that the assumption by principal *B* that the published key is in fact good is contradictory. Though this is allowed by the formalism they claim "though not manifested by our formalism, it is not beyond the wit of man to notice. From this absurd assumption, Nessett derives an equally absurd conclusion".

This rejoinder is pretty much to the point! Part of BAN logic folklore.

[28] Michael Burrows, Martin Abadi, and Roger M. Needham. The Scope of a Logic of Authentication. Revised Research Report 39, Digital Systems Research Center, 1994.

This is intended as an annex to the original BAN report [26].

[29] C. C. I. T. T. Recommendation X.509. *The Directory-Authentication Framework*. C. C. I. T. T., December 1988.

This contains draft proposals for various protocols.

[30] E. A. Campbell, R. Safavi-Naini, and P. A. Pleasants. Partial Belief and Probabilistic Reasoning in the Analysis of Secure Protocols. In *Proceedings 5th IEEE Computer Security Foundations Workshop*, pages 84–91. IEEE Computer Society Press, 1992.

A rather interesting paper; there is clearly work to be done in this (or related) areas. The paper describing a formal system in which elements of logic have probabilities associated with them. This allows the real world to be modelled more accurately. Logical deductions depend on the correctness of such elements and so are associated with probabilities. The paper finds bounds on various probabilities of interest using linear programming methods. Some examples are given.

Well worth a read.

[31] Ulf Carlsen. Cryptographic Protocol Flaws. In *Proceedings 7th IEEE Computer Security Foundations Workshop*, pages 192–200. IEEE Computer Society, 1994.

This paper presents a categorisation of protocol flaws. The categories are:

**Elementary flaws** some protocols may provide only marginal protection, e.g. ones that communicate passwords in the clear, or the Nessett counterexample [89] to the BAN analysis approach.

**Password guessing flaws** (passwords may be used to generate keys, and the practical limitations of such approaches allow a brute force but biased search)

**Freshless flaws** identified by the inability of one principal to detect whether a message has been created recently or not. The Denning-Sacco attack [42] on the conventional key Needham Schroeder protocol is a famous example. Burrows Abadi and Needham demonstrate a freshness flaw in the Andrew Secure RPC protocol [26].

**Oracle flaws** a principal inadvertently acts as a decryption agent for a penetrator. There are examples of single-role and multi-role oracle attacks (i.e. when a principal is limited to one role or may take part in several roles). The three-pass protocol of Rivest, Shamir and Adleman is the subject of these oracle attacks. The three-pass protocol has the following steps:

$$\begin{array}{lll} (1)\ A \rightarrow B & : & E(Ka : M) \\ (2)\ B \rightarrow A & : & E(Kb : E(Ka : M)) \\ (3)\ A \rightarrow B & : & E(Kb : M) \end{array}$$

After receiving the message at line 2 $A$ decrypts with key $ka$ and assuming commutativity sends the result back to $B$ in line 3. The single role oracle flaw is for the intruder simply to pretend to be $B$ and return $\{M\}_{ka}$ back to $A$ in line 2 and hence obtain $M$ in line 3. The author suggests that a "typing" check (to see whether the third message is really an encrypted one) might solve this problem. As presented there is a more obvious problem that shows that this will not work, namely there is nothing to stop the intruder simply acting as $B$ using a key $kc$ that she knows in place of $kb$. The protocol then works as normal. *Have I missed something?*

**Type flaws** a subclass of oracle flaws where in addition to using a fragment of the protocol as an oracle, the penetrator exploits the inability of at least one principal to associate a word (or message) with a particular state of a particular protocol. Five different "types" of information can be distinguished:

- cryptographic protocol;
- protocol run;
- transmission step;
- message (sub) components;
- primitive types.

The paper then gives example type flaws exposed in the Neuman-Stubblebine protocol and the Otway-Rees protocol.

**Internal Flaws** these are due to a principal failing to carry out the necessary internal actions correctly (e.g. failing to make a check). The paper states that a common feature of many formal and semi-formal cryptographic protocol specification methods is their lack of stating internal actions.

**Cryptosystem related flaws** these arise due to the interactions of protocols and the particular cryptographic methods employed.

The paper is well-written and very useful. Of particular note is the discussion of type flaws.

[32] Ulf Carlsen. Generating Formal Cryptographic Protocol Specifications. In *Proceedings 7th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 1994.

The traditional method of specifying protocols has a well-defined syntax but no semantics. They tend very much to resemble the implementation. In this paper the author provides a means to automatically supply an interpretation by making plausible assumptions. Rules are provided to determine the types of message components (keys, addresses,nonces etc), and to infer assumptions and goals. Internal actions are addressed too (for example, the presence of a word with type "nonce" implicitly indicates that a nonce should be generated. Also, checking of values can be inferred. A tool has been developed that can take a standard notational specification and generate a protocol specification in the CKT5 language. Predicates are created describing the behaviours of each principal, the assumptions and goals for each principal. Overall statements of correctness (of the goals with respect to the assumptions) can then be stated and proved. A very interesting paper since it provides one means for overcoming some of the well-known deficiencies of the standard notation.

[33] Ulf Carlsen. Optimal Privacy and Authentication on a Portable Communications System. *Operating Systems Review*, 28(3):16–23, 1994.

This paper reviews some previous work in the field of portable communication systems (PCSs). Various flaws are exposed and some fixes offered. The paper discusses both initiator and responder (i.e. the other end) protocols. Secret and public key approaches are addressed. The paper is well worth a read (the area is set to become very big). One of the suggested protocols seems flawed (the responder protocol of figure 5 of the paper does not necessarily provide authentication of the RCE to the portable).

[34] John Clark and Jeremy Jacob. On The Security of Recent Protocols. *Information Processing Letters*, 56(3):151–155, November 1995.

In this paper the authors describe some attacks on recently published protocols highlighting assumptions about cipher block chaining use but also a flaw in a (corrected) version of the SPLICE authentication protocol (also independently discovered by Lowe of the Programming Research Group at Oxford).

[35] John Clark and Jeremy Jacob. Attacking authentication protocols. *High Integrity Systems*, 1(5):465–474, August 1996.

This paper provides a summary of ways in which protocols fail and provides many examples of such flaws. Methods of attack include: freshness attacks, type flaws, parallel session attacks, binding attacks and some implementation dependent attacks (e.g. Boyd's bit flipping with stream ciphers and cipher block chaining mishaps). Appraisal is probably best left to the reader!

[36] John Clark and Jeremy Jacob. Non-Repeatability is Not Enough.

A preliminary paper. The authors demonstrate that advice on the use of cipher block chaining is either wrong or the rationale is incomplete. If predictable initial blocks are used then in many cases it will be possible for a principal to create the ciphertext for an arbitrary message of his choice.

[37] John Clark and Jeremy Jacob. Freshness is Not Enough: Note on Trusted Nonce Generation and Malicious Principals.

In this paper the authors demonstrate an unusual attack on a mutual authentication protocol by Woo and Lam [117] described in section 6.3.11. Malicious choice of a nonce by one principal can cause a previously issued key to be accepted as fresh by the other principal.

[38] D. Coppersmith. The Data Encryption Standard (DES) and its strength against attacks. *IBM Journal of Research and Development*, 38(3):243–250, May 1994.

In this paper the author argues that the DES algorithm is remarkably resilient to differential cryptanalytical attacks. This is because the method was known to the IBM designer team in 1974. This should wake the reader up! What is in the public domain clearly lags well behind what is known to Governments and their agents. The criteria for designing the infamous S-boxes are described and discussed.

An essential read for cryptanalysts everywhere.

[39] D. W. Davies and W. L. Price. *Security for Computer Networks*. John Wiley and Sons, 1 edition, 1994.

This is a well-established text in the field covering a variety of network security concepts. It encompasses both theoretical approaches to authentication as well as practical examples. The information is a little dated now but this is still a useful book.

[40] D Davis and R Swick. Network Security via Private-Key Certificates. *Operating Systems Review*, pages 64–67, 1990.

A private key certificate is effectively a ticket published by a server to itself. The ticket contains a key, principal identifier and lifetime. The identified principal may supply the ticket and use the corresponding key until the ticket expires. Various applications are suggested (key translation and key distribution). On close analysis it would appear that two of the suggested protocols can be attacked: the initial key translation protocol makes assumptions about the content of the user supplied component of a message (if it starts with a principal identifier then fraudulent messages can be created using the translator as an oracle). The key distribution between server domains using public key allows one of the servers to masquerade as the other.

[41] D. E. Denning. *Cryptography and Data Security*. Addison Wesley, 1982.

This has become a classic text for introductory cryptography. It covers theory of cryptography and explains the fundamentals of various algorithms before moving on to cover non-communications aspects of security.

[42] Dorothy Denning and G. Sacco. Timestamps in Key Distribution Protocols. *Communications of the ACM*, 24(8), August 1981.

The authors examine first the Needham Schroeder conventional key protocol [87]. Under the assumption that keys cannot be compromised the protocol is regarded as secure. But if a key is compromised then it is shown that a penetrator can fool a principal into accepting that key again (it is worth noting that a malicious initiator, who obtained the key in the first place, can also cause the key to be re-accepted).

However, this problem is removed by the incorporation of timestamps into the protocol messages. This is the most widely cited protocol flaw.

The paper then discusses the use of timestamps in public key systems to distribute public keys and also shared keys. Finally the consequences of the compromise of private keys is addressed.

The protocol to distribute symmetric keys using public keys is flawed but this was discovered only in 1994 by Martin Abadi (see 6.9).

A well-written and very clear technical note. A landmark paper.

[43] Whitfield Diffie. The First Ten Years in Public Key Cryptography. *Proceedings of the IEEE*, 76(5):560–577, May 1988.

An excellent survey of public key cryptography. The paper provides a technical introduction to the various advances in the area (exponential key exchanges, knapsacks, RSA, the rise-fall cycle of knapsacks etc). The paper

than addresses implementation issues and where public key cryptography is going. A good read generally.

[44] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

A classic paper in the field. Heralds the birth of public key cryptography.

[45] *Federal Information Processing Standard 46 – the Data Encryption Standard*, 1976.

This is the prime reference for the Data Encryption Standard. It is well-written and easy to read. The algorithm is of course described elsewhere.

[46] W. Fumy and P. Landrock. Principles of Key Management. *IEEE Journal on Selected Areas in Communications*, pages 785–793, June 1993.

This paper provides an overview of issues involved in key management. It describes security requirements and a hierarchical approach to providing them. It's quite high level but is quite deceptive in its range.

[47] Paul Gardiner, Dave Jackson, Jason Hulance, and Bill Roscoe. Security Modelling in CSP and FDR: Deliverable Bundle 2. Technical report, Formal Systems (Europe) Ltd, April 1996.

This report indicates how algebraic techniques can be incorporated within the CSP/FDR approach to security protocol analysis. Such algebraic manipulation is necessary if the approach is to discover attacks which utilise for example particular properties of encryption (commutativity of encryptions etc.). Implementation details (code) are given in this report. The report describes how algebra may be modelled within an extended form of CSP (that used by FDR2) with results of initial evaluation. Later sections address how the approach can be applied to the analysis of some well-known protocols. Implementation attacks arising due to particular modes of encryption (e.g. CBC, CFB etc) are identified as highly troublesome; the state space becomes enormous very quickly.

[48] Paul Gardiner, Dave Jackson, and Bill Roscoe. Security Modelling in CSP and FDR: Deliverable Bundle 3. Technical report, Formal Systems (Europe) Ltd, July 1996.

This represents a continuation and enhancement of the work reported in [47]. The refined approach is used to detect a well-known flaw in the CCITT protocol. One enhancement is the use of a ”lazy spy” — considering only those behaviours of an intruder which are reachable given the specific history of values observed in a sequence of protocol runs (rather than the whole behaviour space of the intruder).

[49] Dieter Gollman. What do we mean by Entity Authentication? In *Proceedings 1996 IEEE Symposium on Research in Security and Privacy*, pages 46–54. IEEE Computer Society, may 1996.

The goals of authentication provide the basis for much controversy. In this paper Gollman identifies 4 notions of authentication. He points out that some "attacks" on protocols really depend on what you think the protocol is intended to achieve. Examples are given. He maintains that the description of authentication in user-oriented anthropomorphic language may at times be harmful. We need to be aware of the gap between the user-oriented descriptions and the electronic message passing that actually occurs. The paper also addresses the misuse of encryption by protocol specifiers.

[50] Li Gong. Using One-way Functions for Authentication. *Computer Communication Review*, 19(5):8–11, October 1989.

This brief paper presents a mutual authentication algorithm based on the notion of keyed (with passwords) one-way functions. The protocol also effects key distribution. The approach has the benefit that one-way functions are probably easier to create than encryption algorithms since there is no need to ensure invertibility. It is claimed that using one-way functions to develop authentication protocols would not necessarily restrict the capabilities that could be offered.

[51] Li Gong. A Note on Redundancy in Encrypted Messages. *Computer Communication Review*, 20(5):18–22, October 1990.

Redundancy in messages can be used to provide checks that a message has not been modified in transit. Explicit redundancy can be detected by anyone with the correct encryption key. An example would be data concatenated with a checksum and which is then encrypted. A problem is that this provides a means by which an attacker can verify keys he or she has guessed. Protocols that encrypt with weak keys, for example passwords, are vulnerable to a guessing attack. Implicit redundancy can only be recognised by the intended recipient(s) who knows the key for a particular example for a particular exchange. Examples are given.

[52] Li Gong. Handling Infeasible Specifications of Cryptographic Protocols. In *Proceedings of The 4th IEEE Computer Security Foundations Workshop*, pages 99–102. IEEE Computer Society, June 1991.

This paper addresses the issue of specification and analysis of infeasible specifications when the analysis is BAN style [26]. The paper provides an outline of how GNY logic [55] can be amended so that principals can send only messages they can realistically expect to. This is done via the notion of *eligibility*. The method ensures that before a message can be sent, the sender must be in possession of the bit strings to be transmitted and it must hold the beliefs implied by transmission of the message. Inference rules to accommodate the changes are presented.

[53] Li Gong. Variations on the Themes of Message Freshness and Replay and Replay or, the Difficulty of Devising Formal Nethods to Analyse Cryptographic Protocols. In *Proceedings 6th Computer Security Foundations Workshop*, pages 131–136. IEEE Computer Society Press, 1993.

This paper describes a variety of ways in which freshness identifiers may be used. Three parties are identified:

- the supplier who creates the identifier;
- the prover that inserts the identifier in a message; and
- the verifier who establishes the freshness of a message by examining the message composition, especially the use of the freshness identifier.

The paper addresses the use of timestamps, truly random numbers, counters, pseudorandom numbers, synchronised counters and pseudorandom number generators and fresh encryption keys. The paper presents a table indicating which mode of use is secure for a particular freshness approach indicating whether the prover is to be trusted or not. A brief categorisation of message replays is then given.

[54] Li Gong, A. Lomas, R. Needham, and J. Saltzer. Protecting Poorly Chosen Secrets from Guessing Attacks. *IEEE Journal on Selected Areas in Communications*, 11(5), jun 1993.

In some systems the use of weak keys is permitted, for example the use of passwords to encrypt authentication data. An intruder might consider guessing such keys as his best line of attack against the system. For such attacks to work he needs to be able to check whether his guess is correct. The protocol should make such verification impossible. This leads to the concept of *verifiable text*. The authors demonstrate several protocols that use random nonces to mask redundancy that might give rise to verifiability. The paper is unusual and well worth a read.

[55] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning About Belief in Cryptographic Protocols. In Deborah Cooper and Teresa Lunt, editors, *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society, May 1990.

This paper presents an extension to the original BAN logic [26]. Interesting extensions include the notion of recognisability (the use of typing would prevent many identified protocol flaws), the notion of possessions (formulæ that a principal can posses, because he has seen them, and so on). In distinction to BAN-logic a principal does not have to believe in a formula in order to include it in a message (he merely has to possess it). Also included are explicit "not-originated here" indications for message components allowing the principal to detect replays of messages he himself has created. There are also message extensions by which preconditions for actually sending a message are attached to it. The derivation rules given are much more numerous (over 40) than those given in BAN.

[56] John Gordon. Public Key Cryptosystems. In *Proceedings of Networks '84*, 1984.

This is an introduction to public key cryptography for the beginner. The paper is very light but there are some helpful analogies for the non-cognoscenti.

The paper gives an outline of one-way functions, trapdoors, key exchange, public key approaches and electronic signatures.

In general little maths is assumed (indeed the notions of modular arithmetic are explained for example) and many of the details are glossed over (e.g. on explaining the disguising of super-increasing sequences in the knapsack problem). Merkle-Hellman and RSA schemes are outlined very briefly.

[57] R. Hauser, P. Jansen, R. Molva, G. Tsudik, and E. van Herreweghen. Robust and Secure Pasword and Key Change Method. In Dieter Gollmann, editor, *Computer Security—ESORICS '94*, number 875 in Lecture Notes in Computer Science, pages 107–122. Springer, 1994.

This paper addresses the problem of how passwords can be changed in a distributed environment and in the presence of failures (for example, acknowledgement messages not getting through).

There is a description of Kerberos (V4 and V5) CHANGEPW and a critical examination (note that there is a typographical error in the V4 description). The question is then raised as to what happens in the protocol when failures occur.

A robust solution to the password update problem is then provided. It assumes that a user who does not successfully complete a transaction (from his point of view) will repeat attempts to change the password in the same way. Effectively the request message contains tickets with the old (new) password encrypted with the new (old) password (it's more complex than this). The authentication server will either store the "old" password or the "new" password with the user depending on whether the previous attempt succeeded or not. Decrypting an appropriate ticket ought to give sufficient proof of identity. If the server already has carried out the update then there is no change and a success response is given, otherwise the update process goes ahead. Switching the order of the tickets (which would allow an intruder to reverse the change) is protected against by the inclusion of nonces (and functions thereof) to ensure that the ordering of the messages is determinable (one is actually a timestamp).

It is hard to say how much of a problem this paper addresses. Intuitively it would seem far from "crucial" as stated by the authors. The proposed solution is quite neat though and certainly it seems efficient. Worth a read.

[58] Martin E Hellman. The Mathematics of Public Key Cryptography. *Scientific American*, pages 130–139, August 1979.

This article provides a very good introduction to public key cryptography mathematics. The author addresses general principles such as NP-hardness and provides an explanation of knapsack and RSA approaches. The mathematics is described well and many simple examples are given. A very good place to start.

[59] Tzonelih Hwang and Yung-Hsiang Chen. On the security of SPLICE/AS: The authentication system in WIDE Internet. *Information Processing Letters*, 53:97–101, 1995.

This paper presents two attacks on the SPLICE/AS authentication protocol. The flaws are caused by signing after encryption. Solutions are offered to fix the flaws. Clark and Jacob [34] show that there still remains a flaw.

[60] Tzonelih Hwang, Narn-Yoh Lee, Chuang-Ming Li, Ming-Yung Ko, and Yung-Hsiang Chen. Two Attacks on Neuman-Stubblebine Authentication Protocols. *Information Processing Letters*, 53:103–107, 1995.

This paper presents two attacks on the Neuman Stubblebine protocol. The first is that given by Carlsen [31] in 1994 (but note that this paper was submitted before Carlsen's was published). The second is a parallel session attack using the one principal as an oracle. Suggestions of how to avoid this are made. The authors are aware of the problem to be solved and in addition to the method shown they suggest some alternatives (such as permuting the order of encryption to avoid replays in different messages). In fact it would appear that this approach is actually more secure since the protocol as it stands could be implemented using cipher block chaining. In that case a replay becomes possible, with the replayed message just an

initial segment of the first message; this wouldn't be the case if there was some plaintext permutation before encryption.

The improvement for the subsequent authentication protocol also depends on implementation for security. Examination shows that if cipher block chaining is used then there is a problem with the improved solution. Thus, the offered solution is implementation dependent.

[61] ISO/IEC. *Information Technology - Security techniques — Entity Authentication Mechanisms Part 1: General Model*, 1991.

This is the introductory part of the ISO/IEC 9798 standard dealing with entity authentication mechanisms . Its basic function is to provide definitions and notation used in the subsequent parts.

[62] ISO/IEC. *Information Technology - Security techniques — Entity Authentication Mechanisms Part 2: Entity authentication using symmetric techniques*, 1993.

This part of the ISO/IEC 9798 standard presents several unilateral and mutual authentication protocols based on shared key cryptography. Advice is given on the use of text fields and also on the choice of time varying parameters(e.g. random, sequence numbers and timestamps).

[63] ISO/IEC. *Information Technology - Security techniques — Entity Authentication Mechanisms Part 3: Entity authentication using a public key algorithm*, 1995.

This part of the ISO/IEC 9798 standard presents several unilateral and mutual authentication protocols based on public key cryptography. They would appear secure at present.

[64] ISO/IEC. *Information Technology - Security techniques — Entity Authentication Mechanisms Part 4: Entity authentication using cryptographic check functions*, 1993.

This part of the ISO/IEC 9798 standard presents several unilateral and mutual authentication protocols based on keyed hash functions.

[65] ISO/IEC. *Information Technology - Security techniques — Entity Authentication Mechanisms Part 5: Entity authentication using zero knowledge techniques*, 1993.

This part of the ISO/IEC 9798 standard presents several unilateral and mutual authentication protocols based on shared key cryptography. Advice on the use of optional text fields and also on the choice of time varying parameters is given too (e.g. random, sequence numbers and timestamps).

[66] A Jiwa, J Seberry, and Y Zheng. Beacon Based Authentication. In Dieter Gollmann, editor, *Computer Security — ESORICS '94*, number 875 in Lecture Notes in Computer Science, pages 125–142. Springer, 1994.

The paper provides a good introduction to Beacon-based authentication. Conventions in cryptography are explained and an outline of Rabin's (the originator) approach to beacon use is given. A beacon emits at regular intervals a random integer in the range 1 to $N$ where $N$ is publicly known. The use of this emitted token is given with respect to the contract signing problem. (How do we solve the problem of one party receiving a commitment from another and yet not being committed him/herself?)

Simplifying, the parties exchange preliminary contracts and commitments to sign (conditional) followed by random integers $I1$ and $I2$. Let $I = (I1 + I2) \bmod N$. They now exchange messages committing them to the contract if the next beacon token is $I$. A party may not commit. If so, he has a 1 in $N$ chance of getting the other party at a disadvantage (and an $N - 1$ in $N$ chance of not getting away with it and having to explain his/her lack of commitment).

A beaconised version of the Needham Schroeder protocol is then given.

The paper is well written and addresses an approach that has not been given much attention.

[67] I Lung Kao and Randy Chow. An Efficient and Secure Authentication Protocol Using Uncertified Keys. *Operating Systems Review*, 29(3):14–21, July 1995.

This presents various repeated authentication protocols. The authors indicate how the use of uncertified keys, i.e. whose validity is not ensured when they are first used may bring performance benefits.

[68] A. Kehne, J. Schöenwälder, and H. Langendörfer. A Nonce-Based Protocol for Multiple Authentication. *Operating Systems Review*, 26(4):84–89, 1992.

A modern repeated authentication protocol . An initial protocol distributes a shared key $K_{ab}$ to principals $A$ and $B$ and also a ticket $\{T, A, K_{ab}\}_{bb}$ sealed by $B$ using a key known only to herself. Repeated authentication can then be carried out by presenting the ticket and using the key $K_{ab}$ (distributed to $A$ by the server under a key shared by it and $A$) and several nonces. After the protocols are presented the BAN logic [26] is used to analyse the protocol. The aim of this protocol is to overcome reliance on accurate distributed clocks (it uses only local clocks for timestamp checks).

There are some problems with the protocol, as pointed out by Syverson [109]. It might be argued that there is a flaw in the idealisation of the protocol. Indeed this is argued by Neuman and Stubblebine [90] that the freshness beliefs in the repeated authentication protocol are invalid. Syverson disagrees with this view and indicates that it is perfectly possible to take another interpretation of "run of the protocol", namely that a run is a single initial protocol and all subsequent runs of the repeated authentication protocol. It would appear that BAN as it currently stands does not handle repeated authentications using tickets.

For the authentication goals they set the authors argue that their initial protocol is minimal with respect to the number of messages.

[69] R. A. Kemmerer. Using Formal Verification Techniques to Analyse Encryption Protocols. In *Proceedings of the 1987 IEEE Symposium on Research in Security and Privacy*, pages 134–139. IEEE Computer Society Press, 1987.

This is one of the first papers to apply logic to the analysis of encryption protocols (rather than algorithms). This is done using a variant of the Ina Jo specification language (and so represents a use of a well-known general specification notation for protocol specification and analysis purposes).

The modern trend has been away from off-the-shelf technologies but this may change.

[70] C. Meadows R. Kemmerer and Jonathan Millen. Three Systems for Cryptographic Protocol Analysis. *Journal of Cryptology*, 7(2):79–130, 1994.

A useful paper. Three systems are described: the Interrogator, the NRL Protocol Analyser and the use of Ina Jo. The tools are used to analyse the TMN protocol with very interesting results. Well worth a read.

[71] V. Kessler and G. Wedel. AUTLOG — An advanced logic of authentication. In *Proceedings of the Computer Security Foundations Workshop VII*, pages 90–99, 1994.

This paper proposes an extension of BAN Logic [26]. It borrows some aspects of existing extensions (e.g. recognisability) but introduces a number of new ones. In particular there is a *recently said* predicate as original suggested by Burrows Abadi and Needham. There is an attempt to simplify the idealisation process by pushing certain aspects of beliefs about keys into the deduction rules. The idealisation process still looks pretty complex though. The authors also introduce the notion of a passive eavesdropper. This can be used to detect certain types of flaw (such as the Nessett flaw

[89]). The paper concludes with discussion about the inability of the logic to handle parallel runs.

[72] Paul C. Kocher. Cryptanalysis of Diffie-Hellman, RSA, DSS, and other Systems Using Timing Attacks. Extended Abstract on PK's WWW Page — http://www.cl.cam.ac.uk/users/rja14/, dec 1995.

A paper that will probably cause quite a stir since it presents an attack on a variety of public key encryption schemes. The attacks are based on noting the amount of time taken to encrypt text. The preliminary results look worrying indeed.

Essential reading!.

[73] Armin Liebl. Authentication in Distributed Systems: A Bibliography. *Operating Systems Review*, 27(4):122–136, October 1993.

This paper provides a brief but wide ranging bibliography of seventy-one papers in authentication. It surveys the field in terms of goals of authentication, design aspects of cryptographic protocols, protocol categorisation (private, public , hybrid, one-ways functions etc.) and verification of protocols. There is a neat table of where to find relevant information on various protocols. The column indicating which protocols are flawed is now out of date!

[74] Gavin Lowe. An Attack on the Needham-Schroeder Public Key Authentication Protocol. *Information Processing Letters*, 56(3):131–136, November 1995.

Seventeen years after publication of the Needham Schroeder Public Key protocol [87] Lowe discovers what everyone else has missed – a parallel session attack. This brief paper is very clear in its descriptions. The flaw was found using the FDR refinement checking tool.

[75] Gavin Lowe. Breaking and fixing the needham schroeder public-key protocol using fdr. In *Proceedings of TACAS*, volume 1055, pages 147–166. Springer Verlag, 1996.

In this paper Lowe describes how the CSP refinement checker FDR was used to identify a hole in the security of the well known Needham Schroeder Public Key Protocol 6.7.1. He presents an account of how principals and intruder communications are modelled in CSP (with a restricted number of principals) and presents an argument to show that the analysis performed is sufficient to guarantee its correctness when more principals are added to the system.

[76] Gavin Lowe. Some New Attacks upon Security Protocols. In *Proceedings of the Computer Security Foundations Workshop VIII*. IEEE Computer Society Pres, 1996.

This paper records a number of attacks on protocols. The aim is largely to show that the same mistakes in protocol design are being made time and time again. The paper contains a more vicious attack on the Woo and Lam Mutual Authentication Protocol than that identified by Clark and Jacob (a public nonce is accepted as a key) (see 6.3.11). This new attack requires a principal to accept messages he has created. Woo and Lam actually state that reflections are detected by principals and so the protocol has no means of enforcing this. Lowe believes that such functionality should be captured by the protocol and not be left as an implementation dependency. Attacks on the KSL protocol 6.5.3 and on the TMN protocol are given.

[77] Gavin Lowe. SPLICE-AS: A Case Study in Using CSP to Detect Errors in Security Protocols. Technical report, Programming Research Group, Oxford, 1996.

In this paper the author indicate show the CSP refinement checker FDR is used to analyse a recently published protocol (which is actually a correction of a previous one).

[78] Wenbo Mao and Colin Boyd. Towards the Formal Analysis of Security Protocols. In *Proceedings of the Computer Security Foundations Workshop VI*, pages 147–158. IEEE Computer Society Press, 1993.

In this paper authors examine some of the defects of BAN logic. After noting that BAN logic passes as secure some patently flawed and insecure protocols they address some specific weakness. First the idealisation process is examined (does not take into account context-specific information), then elements of the nature of belief are examined (such as the senselessness of believing in a nonce). The elicitation of assumptions is also a very difficult area. The authors then go on to provide their own formalism intended to cater for flaws identified. An element of preprocessing is carried out to identify the *implicit* use in the protocol description of various elements (e.g. nonces are identified as challenges, response are identified etc.). A set of BAN-like inference rules are given. Two protocols are then analysed using the system.

[79] Wenbo Mao and Colin Boyd. Development of Authentication Protocols: Some Misconceptions and a New Approach. In *Proceedings 7th Computer Security Foundations Workshop*, pages 178–186. IEEE Computer Society Press, 1994.

This paper addresses two important points regarding authentication protocols. The first is that non-secret data is often encrypted by a principal in order to be retrieved by the intended recipient through decryption. Boyd and Mao argue convincingly that a more desirable way of proceeding is to rely on the one-way service of cryptographic systems rather than the secrecy service. Thus, use of hash functions can be used in order not to provide to much cryptoanalytic information.

The second misconception relates to implementation — the choice of cryptographic algorithm. The authors indicate that the use of cipher block chaining for *all* cryptographic services in authentication protocols may be dangerous and give an example of how a "cut and paste" attack can be mounted on the Otway-Rees protocol.

The third point attacked by the authors is the misuse of redundancy, which can lead to significant provision of cryptoanalytic information.

The authors state that the use of a single notation for all cryptographic services gives a lack of precision that has lead to many weaknesses and provide a notation that distinguishes between encryption for confidentiality and one way services. Their method requires that only secret data be subject to confidentiality encryption.

Overall, the paper is well-written, varied in its scope and very useful.

[80] Wenbo Mao and Colin Boyd. On Strengthening Authentication Protocols to Foil Cryptanalysis. In Dieter Gollmann, editor, *Computer Security—ESORICS 94*, number 875 in Lecture Notes in Computer Science, pages 193–204. Springer, November 1994.

This paper indicates how certain classes of protocols allow a mischiever to generate large amounts of plaintext-ciphertext pairs. A Kerberos description is given and an indication shown that the protocols is subject to an interesting attack.

Suppose a principal sends a request to a server $S$ in the clear. Server $S$ replies with a ticket that includes an encrypted part containing: flag bits, session key, address, names, timestamps, lifetimes, host addresses and authorisation data, all encrypted under the symmetric key $K_{as}$. Now much of this data is known or nearly so. For timestamps etc. the format of the data is known if not the exact data. In the case of an internal adversary $B$ (with whom $A$ wishes to communicate) this is entirely known. However, the amount of data that can be obtained by $B$ in such a way is very small. More important is that the original request is in the clear and so message requests from $A$ can be spoofed ad nauseam. Now what happens if the encryption is done using cipher block chaining? Because the session key will be different in each run, the sequences of ciphertext blocks will be different

too. The way CBC works allows $C_i$ and $P_i + C_{(i-1)}$ to be ciphertext plaintext pairs. Since the session key is different in each run, this merely ensures that different runs allow different plaintext ciphertext pairs to be created.

A description of the KryptoKnight authentication system is then given. This too is subject to an attack generating plaintext ciphertext pairs. Remedies are provided. In one a nonce is exchanged encrypted which then forms part of the MAC generation key. There is effectively a one-time channel and so only one plaintext ciphertext pair is possible on each run. Another scheme using exponentiation key exchange is given.

[81] James L. Massey. An Introduction to Contemporary Cryptology. *Proceedings of the IEEE*, 76(5):533–549, May 1988.

A very good introduction to the history, terminology and theory of cryptology. The author describes both secret key and public key cryptography. The reader will need some mathematics to follow the text. the paper is unusual in that it also attempts to introduce the underlying information theoretic concepts to the reader as well as the more usual algorithm fare.

[82] Cathy Meadows. Formal verification of cryptographic protocols: A survey. In *Proceedings of Asiacrypt 96*, 1996.

Meadows provides a comprehensive account of the use of formal techniques in the development of security protocols. As well as given a survey she also indicates where formal techniques are lacking in use. For example, the use of formal techniques in the design of protocols (i.e. from specification) and also at very low levels.

[83] Jonathan Millen. The Interrogator User's Guide. Technical Report M 93B0000172, MITRE, 202 Burlington Road, Bedford, MA 01730-1420, may 1994.

The Interrogator plays an important part in the development of tool support for protocol analysis. The user specifies the protocol in a prolog-based syntax and can use the tool interactively to determine whether specific states can be reached or specific data items compromised. Its use is illustrated via examples (the Needham Schroeder conventional key distribution protocol, the Diffie Hellman key Exchange and the TMN protocol). the tool provides an automatic search facility but informed guidance from the user is need to reach the stage where automated support is appropriate.

An important tool.

[84] J. K. Millen, S. C. Clark, and S. B. Freedman. The Interrogator: Protocol Security Analysis. *IEEE Transactions on Software Engineering*, 13(2):274–288, February 1987.

This paper gives a description of the Interrogator tool. The user guide [83] is a better place to find detailed information.

[85] Judy H. Moore. Protocol Failures in Cryptosystems. *Proceedings of the IEEE*, 76(5), May 1988.

The use of particular algorithms in conjunction with particular protocols can have disastrous results. This is not because of any inherent weakness in the cryptoalgorithms themselves, rather it is because the way in which they are used requires that they possess certain properties which they do not in fact have. It is the security of the whole system that must be considered not just the algorithm or the protocol in isolation. The paper provides several ways in which features of RSA render its indicated use dangerous: a notary protocol is given in which it is possible to forge a signature on data because exponentiation preserves the multiplicative structure; common modulus and low exponent protocol failures are explained; a low entropy example is cited (the failure arises because of the high redundancy of human speech). Finally various symmetric key failures are identified. The paper is useful in that it highlights the difficulties in going from specification to implementation. All too often specifiers do not state the precise qualities they demand of cryptoalgorithms.

This is an important paper. The whole area of cryptoalgorithm - protocol interaction badly needs addressing (still).

[86] Louise E. Moser. A Logic of Knowledge and Belief about Computer Security. In J Thomas Haigh, editor, *Proceedings of the Computer Security Foundations Workshop III*, pages 57–63. IEEE, Computer Society Press of the IEEE, 1989.

This paper provides a brief overview of the development of modal logics and their use in reasoning about authentication protocols. It introduces a new logic that combines a monotonic logic of knowledge and belief augmented by a non-monotonic **unless** operator. For beliefs of the for "$B(p)$ **unless** $B(q)$" $B(p)$ is assumed to be true unless refuted by other evidence. An example application of the logic is given. Principals wishing to communicate ask a server for a key to be distributed. A characterisation of knowledge and belief about the protocol is recorded in 18 axioms. These axioms encompass rules about beliefs as a result of sending and receiving of messages, knowledge of principals' keys and belief in their security, and trustworthiness of principals. The logic takes the view that belief in a proposition $p$ is presumed unless it is refuted. This has some interesting consequences—for example, should we assume that each principal believes that "$k$ is a key" for all $k$ (Axiom 1d begins to look strange in this context).

A few example consequences of the axioms are then given. The text concludes that there is a need for quantification to be included in the logic. Future plans include also the use of nested **unless** predicates. Combining the logic with a temporal logic is the final suggestion. Currently no tool support is available for the logic (but the authors consider it essential).

The characterisation seems rather complex and as indicated above the actual axioms might usefully be examined. But there is a logic at work here with a sound semantics. An important paper, and worth a read.

[87] Roger Needham and Michael Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12), December 1978.

One of the classic authentication papers. In this paper the authors address issues of establishing interactive communication between principals, authenticated one-way communication (for example, mail systems) and signed communication.

Two protocols for establishing interactive communication are presented: one using conventional symmetric key encryption and the other using public key encryption. The former contains the classic replay flaw[42]. It is is usually referred to as "The Needham Schroeder Protocol" but this could equally apply to any of the three protocols presented in this paper. The authors were aware of many possible attacks and provide a rationale for their protocols. The public key protocol has recently been shown to be susceptible to a parallel session attack by Gavin Lowe. The final protocol described is a means of obtaining digital signatures via a third party using symmetric key encryption.

The authors assume that keys are *not readily discoverable by exhaustive search or cryptanalysis*. As we were later to find out, more restrictive assumptions would be needed.

The paper ends with the well-known quote:

> *Finally, protocols such as those developed here are prone to extremely subtle errors that are unlikely to be detected in normal operation. The need for techniques to verify the correctness of such protocols is great, and we encourage those interested in such problems to consider this area.*

One of the landmark papers in authentication. Essential reading.

[88] Roger M. Needham and M. D. Schroeder. Authentication Revisited. *Operating Systems Review*, 21(7):7–7, January 1987.

In this paper the authors revisit their famous conventional (symmetric) key authentication protocol and show how an extra exchange between the two authenticating principals can be used to overcome the freshness deficiency identified by Denning and Sacco (who overcame the problem by the use of timestamps) [42]. The extra exchange includes a nonce from the second principal *B* to be provided to the authentication server. This is then included by the authentication server in the authentication ticket passed to *B* as part of the protocol, thereby ensuring freshness.

[89] Daniel M. Nessett. A Critique of the Burrows, Abadi and Needham Logic. *ACM Operating Systems Review*, 24(2):35–38, April 1990.

This brief paper (and its rejoinder) formed the start of what might be described as "BAN wars" — the debate over what Burrows, Abadi and Needham claimed for BAN logic (first order claim) [26], what others claimed they claimed (second order claim) and what the capabilities of BAN logic and its derivatives are (no claims, just investigative science).

Nessett misquotes (or misinterprets) the BAN authors statements regarding goals of authentication. This paper implies that the BAN authors had a particular position on what the belief goals of a protocol should be. This is simply wrong; the BAN authors give the indicated goals merely as examples of what might be suitable in particular circumstances. Indeed, the BAN authors even describe the Otway Rees protocol is a "well designed protocol that may have use in certain environments", even though the protocol does not achieve the goals Nessett states they regard as necessary (a point raised by Syverson [107]).

The more important point of this paper is that it provides an example protocol that is obviously insecure but the flaw is not detected by BAN analysis. The crux of the example is that a principal can broadcast a message that contains a key for shared use and a nonce all encrypted with her private key. This is obviously readable by everyone (with the public key) and so the protocol is insecure. The protocol given is sufficient to establish first and second order beliefs of both parties in the goodness of the key.

The paper is now part of authentication folklore.

[90] B. Clifford Neuman and Stuart G. Stubblebine. A Note on the Use of Timestamps as Nonces. *Operating Systems Review*, 27(2):10–14, April 1993.

This paper gives an alternative protocol similar to that given by Kehne *et al* [68] for repeated authentication. The ticket is slightly different to the one used in that protocol. In addition, although timestamps are still local, the ticket is sealed by the authentication server, rather than by one of principals.

The paper criticises the application of BAN logic [26] to the KLS protocol [68], stating that it violates the notion of freshness. Further modifications to this protocol are suggested. The tradeoffs involved in adopting particular approaches to authentication (e.g. timestamps or nonces) are examined.

The protocol has certain flaws as exposed by Hwang et al [60].

[91] D. Otway and O. Rees. Efficient and Timely Mutual Authentication. *Operating Systems Review*, 21(1):8–10, January 1987.

This paper presents the well-known Otway-Rees protocol. The notation is a little different from usual. The description is of interest for historical reasons: the protocol has become one of those regularly subject to analysis techniques (e.g. BAN [26]). There are some classic replay flaws in the protocol.

[92] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

The authors present the RSA algorithm for the first time in an academic journal (it had appeared previously in one of Martin Gardner’s columns). A classic paper.

[93] A.W. Roscoe. Intensional Specifications of Security Protocols. In *Proceedings 9th IEEE Computer Security Foundations Workshop*, pages 28–38. IEEE Computer Society Press, 1996.

The author introduces two notions of specification: *extensional* and *intensional*.

An extensional specification indicates what the protocol is to achieve (but not how). Given a set of assumptions prior to the protocol about ”states of mind” of the principals an extensional specification will give what properties of those states of mind of the principals must hold after execution of the protocol. The author illustrates how such specifications may be found lacking if attacks are to be found. An intensional specification describes properties of how communications between principals must occur. It does not specify precisely what is achieved. A deviation from the designer’s intended sequence of communications is an attack. Other methods must be applied to address what the protocol actually achieves (e.g. BAN analysis). Coding such specifications in CSP allows the refinement checker FDR to be brought to be to search for deviational attacks. Well worth a read. The work was carried out under the Strategic Research Plan managed by the DERA.

[94] Bill Roscoe and Paul Gardiner. Security Modelling in CSP and FDR: Final Report. Technical report, Formal Systems Europe, oct 1995.

This report summarise how CSP can be used to model principals' behaviour in security protocols. Authentication is couched as a refinement problem and the refinement checker FDR is used to carry out a state space exploration to determine whether a proposed 'implementation' actually satisfies the specification of authentication. This report is the initial output of the Formal Systems work indicating that CSP/FDR could be a very promising means of analysing security protocols. The work was carried out under the Strategic Research Plan managed by the DERA.

[95] A. D. Rubin and P. Honeyman. Formal Methods for the Analysis of Authentication Protocols. Technical Report Technical report 93–7, CITI, November 1993.

This paper provides a review of extant literature in the field of authentication protocols. The paper begins with some introductory definitions and a description of the Needham-Schroeder protocol [87], its flaws and their resolution by timestamps. This serves as a good motivation for the subject matter that follows. The various approaches to analysis are then investigated. The reference material is wide-ranging and the text is well written.

[96] Peter Ryan. The Design and Verification of Security Protocols. Technical report, Defence Evaluation Research Agency, August 1996.

This report provides a readable summary of the modelling of authentication protocols using CSP and how the FDR tool can be used to verify that a protocol is secure. The approach models principals, a general intruder and the network medium as communicating CSP processes. The approach has the merit that several themes of security can be addressed (confidentiality, integrity, availability, etc). Examples of where the approach has discovered new attacks are given. The considerable promise shown by the approach is indicated but at the same time current limitations and possible developments are clearly addressed and discussed.

[97] Steve Schneider. Security Properties and CSP. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 174–187. IEEE Computer Society Press, may 1996.

This paper addresses issues how confidentiality and authenticity of messages can be addressed within the CSP framework. The paper provides a brief introduction to CSP and can be read without prior knowledge of the process algebra.

[98] B. Schneier. The IDEA Encryption Algorithm. *Dr. Dobb's Journal*, pages 50–56, December 1993.

This article provides a good, easily understood description of the IDEA conventional key encryption algorithm. This is a 64-bit block algorithm with a 128-bit key. The main diagram seems slightly out of kilter with the text though.

Software implementations are about 1.5 to 2 times as fast as corresponding DES implementations. The author cites a VLSI implementation that encrypts at 177MBits/s when clocked at 35MHz.

[99] Bruce Schneier. *Applied Cryptography*. Wiley, 1994.

Probably the best available introductory text on modern day cryptography and its applications. It is easy to read and very wide ranging in the topics covered. Many conventional key and public key algorithms are described together with known weaknesses. A significant amount of effort is expended explaining various cryptographic *protocols*. There are several attacks on protocols described but some attacks that we know about are not covered.

[100] G.J. Simmons. How to Insure that Data Aquired to Verify Treaty Compliance are Trustworthy. *Proceedings of the IEEE*, 76(5):621–627, May 1988.

A cold war summary paper! Rather less flippantly, Simmons provides a readable account on the work of treaty verification (for nuclear tests) carried out at Sandia Laboratories. Descriptions are given of both symmetric key and public key approaches.

[101] Miles E. Smid and Dennis K. Branstad. The Data Encryption Standard: Past and Future. *Proceedings of the IEEE*, 76(5):550–559, May 1988.

This is a good paper to read. It examines the history of DES, why it was produced, who were the major stakeholders and how it was taken up by various bodies. An overview of its applications is given. Not much technical information but a good overview of the state of play in 1988.

[102] Einar Snekkenes. Exploring the BAN Approach to Protocol Analysis. In *1991 IEEE Symposium on Research in Security and Privacy*, pages 171–181. IEEE Computer Society Press, 1991.

Snekkenes shows that BAN logic is incapable of detecting errors due to permutation of protocol steps. He also shows that it is unlikely that a BAN type approach can hope to provide good analysis of zero-knowledge type protocols. A class of protocols is introduced called *terminating* protocols and it is shown that Dan Nesset's flawed protocol [89] belongs to this class.

[103] E. Snekkenes. Roles in Cryptographic Protocols. In *Proceedings of the 1992 IEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1992.

In this paper Snekkenes shows that Bieber's approach to protocol verification [16] may not detect flaws that arise due to principals taking on more than one role in a protocol. Bieber's logic may, however, be successfully modified. An example protocol is given that is deemed secure by both BAN analysis and Bieber CKT5 analysis.

[104] Einar Snekkenes. *Formal Specification and Analysis of Cryptographic Protocols*. PhD thesis, Faculty of Mathematics and Natural Sciences, University of Oslo, Norwegian Defence Research Establishment, P.O. Box 25, N-2007, Kjeller, Norway, jan 1995.

This DPhil thesis provides a formal framework for specifying and reasoning about protocols. Numerous theories have been written in HOL to support the analysis of protocols.

[105] Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. Kerberos:An Authentication Service for Open Network Systems. jan 1988.

The most usual academic reference for the early Kerberos. Very clearly written. The paper addresses protocol issues but also administration and application programmer views.

[106] Stuart G. Stubblebine and Virgil D. Gligor. On Message Integrity in Cryptographic Protocols. In *Proceedings of the 1992 IEEE Symposium on Research in Security and Privacy*, pages 85–104. IEEE, 1992.

This paper deals with the possibility of forging messages by cutting and splicing of transmitted messages. The principal mode of encryption considered is cipher block chaining (CBC). The Kerberos authentication protocols is chosen as an example to be attacked in this way. The paper addresses the adequacy (or otherwise) of checksums to protect against such malicious modification. A detailed but very good read.

[107] Paul Syverson. The Use of Logic in the Analysis of Cryptographic Protocols. In Teresa F. Lunt and John McLean, editors, *Proceedings of the 1991 IEEE Symposium on Security and Privacy*, pages 156–170. IEEE Computer Society, May 1991.

The author is concerned with placing the use of logics for analysing security protocols on a formal footing. In particular there is a worry that the capabilities and limitations of particular logics are not really understood and that some tool is needed to allow analysis of the logics. He proposes *possible worlds semantics* as that tool.

**Section 2 : Security Trust and Intentionality** The author categorises the objectives of protocol analysis logics for both epistemic (knowledge) and doxastic logics (belief). Loosely, belief is concerned with trust, functionality and a legitimate subject's point of view, whereas knowledge logics are used to investigate security and a penetrator's viewpoint.

Though intuitively epistemic logic seems more appropriate for security and doxastic logics for trust, each is formally capable of capturing and reasoning about trust and security. Practical concerns about the amount of work involved in doing proofs lead the author to recommend that future research concentrate on epistemic logics.

**Section 3: BAN Logic** This section provides a critique of BAN logic [26] and also comments on others' critiques of BAN.

Syverson points out that while the BAN authors themselves have a good idea of what they are doing, others are occasionally confused about the authors' goals. In particular the goals of authentication are a considerable source of confusion. Nessett's criticisms [89] are examined. The author correctly points out that the BAN authors take no position on the goals of authentication, rightly considering these goals to be application specific.

One of the BAN authors' statements "common belief in the goodness of *K* is never required—that is, *A* and *B* need not believe that they both believe that. . . that they both believe that *K* is good" is criticised, because such demonstration is known to be impossible in general. He goes on to give an example where second order beliefs are insufficient and concludes that the degree of belief demonstrated by a protocol varies according to the application.

The author also takes to task Cheng and Gligor for several misattributions. He also examines Nessett's claims and points out that in places the original BAN paper might give the impression of handling (at least some) security issues. He refers to the table of protocols included in the BAN paper but indicates that the authors of BAN included these 'bugs' as "aspects our formalism helped bring to light".

It is maintained that BAN logic has been much misinterpreted and that in practice it has helped reveal several flaws. As a *formal* method, however, the author does not support the use of BAN logic.

**Section 4: Semantics** The author examines the role of semantics. One of the major roles of a semantics is to give a means of evaluating logics. Generally, we would want to show soundness and completeness. Although soundness is often the principal concern, for security applications completeness is seen as being of "utmost importance". A formal semantics provides a precise structure with respect to which such completeness and soundness can be proven. The author argues that if a semantics takes its structure directly from the logic, then no assurance is gained about the adequacy of the logic from soundness or completeness theorems (indeed they should be trivial). A further view on formal semantics is that it supplies an alternative view (diversity).

The author introduces possible world semantics offering this as a means of exposing the Nessett flaw.

Overall: The paper is well written and is well worth a read.

[108] Paul Syverson. A Taxonomy Of Replay Attacks. In *Proceedings of the 7th IEEE Computer Security Foundations Workshop*, pages 131–136. IEEE Computer Society Press, 1994.

This paper presents a taxonomy of replay attacks; or rather, two taxonomies: an origination taxonomy (based on the protocol run of origin of replayed message) and a destination taxonomy (based on the recipient of the replayed message relative to its intended recipient). Origination splits replays into *run-external* attacks (replay of messages from outside the current protocol run) and *run-internal* ( replay of messages from inside the current protocol run). Run-external attacks are further divided into interleavings (requiring contemporaneous protocol runs) and classic replays (not necessarily requiring contemporaneous runs).

Within the origination taxonomy can be placed the destination taxonomy: *deflections* (message is sent either to a sender —a reflection— or to a third party) and *straight replays* (intended recipient receives the message but it is delayed).

The paper goes on to describe how the taxonomy provides a framework in which to discuss countermeasures ' capabilities and how it highlights the capabilities of various logical analysis approaches (for example, BAN is generally directed at classic replays). Some examples of replay attacks are given on the BAN-Yahalom protocol.

The taxonomy actually applies to message fragments rather than messages. In practice it may be necessary to use more than one type of replay to mount a successful attack.

The paper is worth a read. There is little startling but that is the way with taxonomies. Those presented in this paper appear useful. The emergence of

useful taxonomies indicates perhaps that protocol development and analysis has come of age?

[109] Paul Syverson. On Key Distribution for Repeated Authentication. In *Operating Systems Review*, pages 24–30, 1994.

This paper describes the two-part Neuman Stubblebine protocol and then shows how it can be attacked. The attacks assume certain implementation dependencies (for example, that the substitution of a nonce for a key will go undetected and that direction bits are not used). After a discussion of countermeasures the paper then presents a variant of the Neuman Stubblebine protocol, which is free from the previous attacks, but then shows how it itself can be attacked. A final protocol that incorporates elements of the KSL protocol [68] is then presented. The paper concludes with an analysis of what the goals of the KSL and NS repeated authentication protocols were and of the utility of BAN logic [26] for addressing repeated authentication.

A good paper, with some nice attacks.

[110] Paul Syverson and Catherine Meadows. A Logic language for Specifying Cryptographic Protocol Requirements. In *Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy*, pages 165–177. IEEE Computer Society Press, May 1993.

As yet unread. Here for completeness.

[111] Gene Tsudik. Message Authentication with One-Way Hash Functions. *Operating Systems Review*, 22(5):29–38, 1992.

This paper assesses the merits of two approaches to using hash functions to provide message authentication: the secret prefix and secret suffix methods. The paper proposes a useful hybrid.

[112] Victor L. Voydock and Stephen T. Kent. Security Mechanisms in High-Level Network Protocols. *Computing Surveys*, 15(2):135–171, June 1983.

An early protocol security classic. This paper describes attacks on communications protocols and measures that can be taken to counter them. Best of all is the low level detail on cryptosystem usage (in particular, the consequences of the use of particular approaches to the choice of initialisation vectors in DES). Essential reading.

[113] *Efficient DES Key Search*, Crypto 93, August 1993.

This paper addresses in considerable detail a design for a pipelined key-search machine for DES. A very good paper. Probably the most detailed hardware paper to reach a conference ever!

[114] Michael Willet. Cryptography Old and New. *Computers and Security*, Vol 1:177–186, 1982.

This is a simple introduction to cryptography assuming no maths whatsoever. It gives a good introductory account to the history of cryptography and introduces various types of encryption algorithm. It is of course a little dated. It takes the reader from Caesar ciphers to DES and public key cryptography (but no details on the latter).

[115] Michael willet. A Tutorial on Public Key Cryptography. *Computers and Security*, pages 1–20, 1982.

Willett provides a very brief overview of mainstream public key cryptography (concentrating on RSA and Merkle Hellman Knapsacks).

[116] T. Y. C. Woo and S. S. Lam. Authentication for Distributed Systems. *Computer*, 25(1):39–52, January 1992.

This paper provides a good introduction to some principles of authentication, explaining some basic cryptography, what the threats to a system are, what sorts of parties may wish to carry out authentication exchanges. The paper provides some paradigms of authentication exchanges. Two case studies (Kerberos and SPX) are given. There are some errors in the paper. Woo and Lam published some corrections shortly after this paper was published indicating that figure 5 on page 47 needs augmenting: the principal $P$ needs to be included to precede the principal $Q$ in steps 5 and 6. Some of the protocols shown are susceptible to attack nevertheless. Indeed, Woo and Lam themselves have published a correction to another of the protocols [117].

[117] T. Y. C. Woo and S. S. Lam. A Lesson on Authentication Protocol Design. *Operating Systems Review*, pages 24–37, 1994.

A previous paper by the authors [116] described a protocol that was subsequently found to be flawed. The authors explain how they started with a secure (but elaborate) one-way authentication protocol and progressively simplified it to take out what was regarded as superfluous information. The simplification steps are given and the transition to insecurity is identified. The authors give a *Principal of Full information*, which dictates that the initiator and responder include in every outgoing message all of the information that has been gathered so far in the authentication exchange. A number of simplification heuristics are given. These are demonstrated by application to a mutual authentication protocol. It would appear that there is a problem with the description given in this paper. The transition to insecurity occurs in the step before the one identified by the authors. Effectively a parallel session attack can be mounted to enable a malicious agent to start and complete an authentication exchange with a server without the principal whose identity he claims knowing that such an authentication has taken place.

[118] S. Yamaguchi, K. Okayama, and H. Miyahara. Design and Implementation of an Authentication System in WIDE Internet Environment. In *Proceedings of the 10th Regional Conference on Computers and Communication Systems*, 1990.

The original description of the SPLICE Authentication System. See [34].

[119] Alec F Yasinsac and William A Wulf. A Formal Semantics for Evaluating Cryptographic Protocols. The paper has been superceeded by a later version published in the IEEE Symposium on Security and Privacy 1994., 1993.

This paper provides a good introduction to some relevant issues for authentication protocols. It provides an overview of the historic development of protocols. After supplying an appraisal (albeit brief) of the capabilities of current approaches to protocol verification the authors go on to suggest an approach based on the notion of weakest precondition calculus. They say:

> Our review of the problem of protocols verification has brought us back repeatedly to the field of program verification. Actions of principals in programs can be thought of as analogous to the operations of programs.

The general idea is that protocol steps are viewed in terms of their results, i.e. they are effectively state transformers, with the sending and receiving of messages modelled as memory accesses.

A language CPAL (Cryptographic Protocol Analysis Language) is given in which the goals of authentication can be specified. Whereas BAN logic models the evolution of principals' beliefs, the aim of the current paper is to model the *actions* a user can take. CPAL provides a language to describe those actions (send/receive messages on a network, encryption and decryption, creating keys, timestamps and nonces, cox/different computing functions and making comparisons and simple decisions). Note that the approach taken is that a protocol does not require a notion of looping (effectively only assignment and alternation are needed).

With the simplified execution model the idea is that the goals of the protocol are stated as a postcondition and then wp-calculus is used to derive the preconditions for success of the protocols (i.e. the initial assumptions).

The appendix to this paper gives a description of the CPAL language and an indication of its use to specify various protocols (Needham and Schroeder Private Key Protocol, Denning and Sacco Private Key Protocol, and the Otway and Rees Private Key Protocol).

The ideas expressed in this paper are useful but the particulars seem a little light at the moment.

[120] Paul C van Oorschot. An Alternate Explanation of two BAN Logic 'failures'. In Tor Helleseth, editor, *Eurocrypt '93*, number 765 in LNCS, pages 443–447. Springer Verlag, 1993.

This paper provides a retort to Boyd and Mao's paper at the same conference discussing limitations of BAN logic approaches [23]. Oorschot maintains (with justification) that BAN passes the first protocol simply because the formal assumption of trust in the authentication server is not actually true. The second example protocol is passed by BAN because the idealisation is simply wrong.