<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.1 Background</strong></td>
<td>5</td>
</tr>
<tr>
<td><strong>1.2 Protocol Source</strong></td>
<td>6</td>
</tr>
<tr>
<td><strong>2.1 General Principles</strong></td>
<td>7</td>
</tr>
<tr>
<td><strong>2.2 Symmetric Cryptography</strong></td>
<td>8</td>
</tr>
<tr>
<td>2.2.1 Classical Cryptography</td>
<td>8</td>
</tr>
<tr>
<td>2.2.2 Modern Cryptography</td>
<td>9</td>
</tr>
<tr>
<td>2.2.3 Mode of Block Cipher Usage</td>
<td>12</td>
</tr>
<tr>
<td><strong>2.3 Public Key Cryptography</strong></td>
<td>13</td>
</tr>
<tr>
<td><strong>2.4 One-way Hash Algorithms</strong></td>
<td>15</td>
</tr>
<tr>
<td><strong>2.5 Notational Conventions</strong></td>
<td>16</td>
</tr>
<tr>
<td><strong>3.1 Symmetric Key Without Trusted Third Party</strong></td>
<td>17</td>
</tr>
<tr>
<td><strong>3.2 Symmetric Key With Trusted Third Party</strong></td>
<td>18</td>
</tr>
<tr>
<td><strong>3.3 Public Key</strong></td>
<td>20</td>
</tr>
<tr>
<td><strong>3.4 Hybrid Protocols</strong></td>
<td>22</td>
</tr>
<tr>
<td><strong>3.5 Other Forms of Protocol</strong></td>
<td>22</td>
</tr>
<tr>
<td><strong>3.6 General</strong></td>
<td>22</td>
</tr>
<tr>
<td><strong>4.1 Freshness Attacks</strong></td>
<td>23</td>
</tr>
<tr>
<td><strong>4.2 Type Flaws</strong></td>
<td>23</td>
</tr>
<tr>
<td><strong>4.3 Parallel Session Attacks</strong></td>
<td>26</td>
</tr>
<tr>
<td><strong>4.4 Implementation-Dependent Attacks</strong></td>
<td>28</td>
</tr>
<tr>
<td>4.4.1 Stream Ciphers</td>
<td>29</td>
</tr>
<tr>
<td>4.4.2 Cipher Block Chaining</td>
<td>29</td>
</tr>
<tr>
<td><strong>4.5 Binding Attacks</strong></td>
<td>33</td>
</tr>
<tr>
<td><strong>4.6 Encapsulation Attacks</strong></td>
<td>34</td>
</tr>
<tr>
<td><strong>4.7 Other Forms of Attack</strong></td>
<td>35</td>
</tr>
<tr>
<td><strong>4.8 Conclusions</strong></td>
<td>36</td>
</tr>
<tr>
<td><strong>5.1 Formal Methods for Analysis</strong></td>
<td>37</td>
</tr>
<tr>
<td><strong>5.2 Notational Conventions</strong></td>
<td>39</td>
</tr>
<tr>
<td>5.2.1 BAN/L</td>
<td>39</td>
</tr>
</tbody>
</table>
5.2.2 Other Logics ......................................... 41
5.3 Expert Systems and Algebraic Writing Systems ............ 42

6 A Library of Protocols .................................. 44
6.1 Symmetric Key Protocols without a Trusted Third Party .... 44
  6.1.1 ISO Symmetric Key One-Pass Unilateral Authentication
       Protocol ........................................ 44
  6.1.2 ISO Symmetric Key Two-Pass Unilateral Authentication
       Protocol ........................................ 44
  6.1.3 ISO Symmetric Key Two-Pass Mutual Authentication .... 44
  6.1.4 ISO Symmetric Key Three-Pass Mutual Authentication .... 45
  6.1.5 Using Non-Reversible Functions ....................... 45
  6.1.6 Andrew Secure RPC Protocol .......................... 45
  6.2 Authentication Using Cryptographic Check Functions ....... 46
  6.2.1 ISO One-Pass Unilateral Authentication with CCFs .... 46
  6.2.2 ISO Two-Pass Unilateral Authentication with CCFs .... 46
  6.2.3 ISO Two-Pass Mutual Authentication with CCFs .......... 46
  6.2.4 ISO Three-Pass Mutual Authentication with CCFs ........ 46
6.3 Symmetric Key Protocols Involving a Trusted Third Party .... 46
  6.3.1 Needham-Schroeder Protocol with Conventional Keys .... 46
  6.3.2 Denning-Sacco Protocol ............................. 47
  6.3.3 Otway-Rees Protocol ............................... 47
  6.3.4 Amended Needham-Schroeder Protocol ................... 48
  6.3.5 Widmer Frog Protocol ............................... 48
  6.3.6 Yahalom .......................................... 49
  6.3.7 Carlse's Secret Key Initiator Protocol ................ 49
  6.3.8 ISO Four-Pass Authentication Protocol ................ 50
  6.3.9 ISO Five-Pass Authentication Protocol ................ 50
  6.3.10 Woo and Lam Authentication Protocols ................ 50
  6.3.11 Woo and Lam Mutual Authentication Protocols .......... 53
6.4 Signature with Conventional Key Encryption ................ 54
  6.4.1 Needham-Schroeder Signature Protocol ................ 54
6.5 Symmetric Key Repeated Authentication Protocols ............ 55
  6.5.1 Kerberos Version 5 .................................. 55
  6.5.2 Neuman Stubbsine ................................. 57
  6.5.3 Kehr Langeendorfer ................................ 58
  6.5.4 The Kao-Chow Repeated Authentication Protocols ....... 58
6.6 Public Key Protocols without a Trusted Third Party ......... 59
  6.6.1 ISO Public Key One-Pass Unilateral Authentication
       Protocol ........................................ 59
| 6.6.2 | ISO Publ i c Ke y Tw o-Pas s Uni l at e r alAut he nt i c at i on Pr ot oc ol | 59 |
| 6.6.3 | ISO Publ i c Ke y Tw o-Pas sMut ualAut he nt i c at i on Pr ot oc ol | 59 |
| 6.6.4 | ISO Thr e e -Pas sMut ualAut he nt i c at i on Pr ot oc ol | 60 |
| 6.6.5 | ISO Tw o Pas sPar al l e lMut ualAut he nt i c at i on Pr ot oc ol | 60 |
| 6.6.6 | Bi l at e r alKe y Exc hange w i t h Publ i cKe y | 60 |
| 6.6.7 | Di f fie He l l m an Exc hange | 60 |
| 6.7 | Publ i cKe y Pr ot oc ol sw i t h Tr us t e d Thi r d Par t y | 61 |
| 6.7.1 | Ne e dham -Sc hr oe de rPubl i cKe y Pr ot oc ol | 61 |
| 6.8 | SPLICE/AS Aut he nt i c at i on Pr ot oc ol | 61 |
| 6.8.1 | H C M SPLICE/AS | 62 |
| 6.9 | Den ni ng Sac c o Ke y Di s t r i but i on w i t h Publ i cKe y | 63 |
| 6.9.1 | CCITT .509 | 63 |
| 6.10 | Mi s c e l l ane ous | 64 |
| 6.10.1 | Sham i rRi ve s tAde l m an Thr ee Pas spr ot oc ol | 64 |
| 6.10.2 | Gong Mut ualAut he nt i c at i on Pr ot oc ol | 64 |
| 6.10.3 | Enc r ypt e d Ke y Exc hange – EKE | 65 |
| 6.10.4 | Davi sSw i c k Pr i vat e Ke y Ce r t i fic at e s | 66 |
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 modeling assumptions about the behavior of other principals to make decisions on how to act. These decisions depend crucially on what validity can be summed 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 distribute secrets to some of those principals or allows some secrets to be recognized [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 secrets (typically cryptographic keys) or that a particular principal is operational. They 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. However, remarkably subtle and many protocols have been shown to be flawed along 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 standard 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 Denning and Sacco [42].
A protocol for e-dham and Schroefer(seventeen years afterward). 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 analyze 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 protocols give little detail 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 goal we also provide a brief review of elements of cryptography.

1.2 A Protocols Resource

Authentication never stops! This report is intended as a compendium of useful information related to authentication. Hopefully, this will be useful to researchers and protocol signers alike. However, the author hopes to make his "living document" and update it as comments from the community are received. The author

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

6
2.1

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 data gram. 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 cipher text or cryptogram. The precise form of the cryptogram corresponding to a plaintext \( P \) depends on an additional parameter known as the key \( K \).

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

\[
\begin{array}{cccccc}
\text{Plaintext} & \rightarrow & \text{Ciphertext} \\
\text{Plaintext} & \leftarrow & \text{Ciphertext}
\end{array}
\]

The classes of encryption and decryption algorithms 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 cipher texts and the ability to determine the plaintexts corresponding to cipher texts.

2.2 Symmetric Key Cryptography

In symmetric key cryptography the encryption key \( K \) and the decryption key \( K \) 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 \( K \) 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 cipher texts corresponding to arbitrary plaintexts and read the contents of messages.

2.1

\[
\begin{array}{cccccc}
\text{C} & \rightarrow & \text{S} & \leftarrow & \text{A} \\
\text{T} & \rightarrow & \text{C} & \leftarrow & \text{T} \\
\text{T} & \rightarrow & \text{C} & \leftarrow & \text{E} \\
1. & \rightarrow & \text{F} & \leftarrow & \text{D} \\
\text{T} & \rightarrow & \text{B} & \leftarrow \\
\end{array}
\]

2.2

\[
\begin{array}{cccccc}
\text{I} & \rightarrow & \text{U} & \leftarrow \\
\text{T} & \rightarrow & \text{O} & \leftarrow \\
\end{array}
\]
2.2.1

Clasical Cryptography

Classical ciphers have 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 cipher text 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, polyalphabetical and polygraphic 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 Will et [114].

2.2.2

Modern day cryptography

Modern day symmetric key algorithms are primarily 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 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 cryptanalytic 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.


1974! O
Ot he re xam pl e sofbl oc k c i phe r sar e MADRYGA (e f fic i e nt for s tw ar e im pl em e nt at i on and ope r at e son 8-bi tbl oc ks ), NEW DES (ope r at e son 64-bi t bl oc ks but w i t h a 120-bi t ke y), FEAL-N, RC2 and RC4 (by Ronal d Rive st ) and IDEA (by Laiand Mass e y). Sc hne i e r has w r i t t e n a r e adabl e ac c ount of the IDEA al gor i t hm [98]. A ve r y good ove r vi e w of bl oc k c i -phercs (and ot he r s ) c an be f ound i n Schne i e r’s ge ne r alc ryp togr aphy t e xt [99].

2.2.3 M odesofBl oc k C i pherUs age

The re ar e s e v e r alm ode s i n w hi c h a bl oc k c i phe r c an be us e d. Pr i nc ip al one sar e :

- **El e c troni Code Book (ECB)**
- **Ci phe rBl oc k Chai ni ng (CBC)**
- **Ci phe rFe e dbac k Mode (CFB)**
- **Out putFe e dbac k Mode (OFB)**

ECB i st he s i m pl e s tm ode . Cons e c ut i ve bl oc ks of pl ai nt e xtr e ar e s i m pl y en c r ypt e d us i ng t he al gor i t hm . Thus , i de nt i c al bl oc ks of pl ai nt extar e al -w a yse nc r ypt e d i n t he s am e w ay (w i t h t he s am er e s ul t ). It ss e c ur i t y ne e ds t o be que s t i one d f or s pe c i fic c ont e xt s . An anal ys tm ay be abl e t o buil d up a c ode book of pl ai nt ext -c i phe r t e xtpai r s(e i t he r know n or be c aus e he c an appl y c r ypt anal yt i c m e t hods t o de r i ve t he pl ai nt ext s ). Al s o, i t i s pos s i bl e t o m odi f y m e s s age s (e .g. by s i m pl y r e pl ac i ng an en c r ypt e d bl oc k w it h anot he r ).

Ci phe rBl oc k Chai ni ng (CBC) i sar e l at i ve l y good m e t hod of en c r ypt i ng se ve r al bl oc ksof dat a w i t h an al gor i t hm f ore nc r ypt i ng a s i n gl e bl oc k. Iti s one m ode i n w hi c h t he w i de l y us e d Dat a Enc r ypt i on St andar d (DES) c an be e m pl oye d. Bl oc k $i$ of pl ai nt e xti s e x cl us i ve l y-or e d (he r e af t e r XORe d) w i t h bl oc k $i-1$ of ci phe r t e xtand i st he n e nc r ypt e d w i t h t he ke ye d bl oc k en c r ypt i on f unc t i on t o f or m bl oc k $i$ of ci phe r t e xt .

For ex am pl e , w i t h i ni t i al i s at i on bl oc k $I$ t he en c r ypt i on of m e s s age bl oc k se que nc e $P_1P_2: : : P_n$ w i t h ke y de not e d by $E(K,P)$ is gi ve n by $E(K,P_1P_2: : : P_n)$ = $C_0C_1C_2: : : C_n$ w he r e $C_0$ = $I_8i$; $i$ > 0

$C_i$ = $e(K:C_{i-1}$$\oplus$$P_i))$
Here, $e(K)$ is the block encryption function used with key $K$. The encryption process is shown in Figure 2. Successive cipher text blocks are decrypted using the key block function $d(K)$ according to the rule $8i > 0 \Rightarrow P_i = C_i \oplus d(K)$.

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

If we choose a different initial block in each case then even identical plain text messages will have different cipher texts. It is widely accepted that non-repeating initial blocks are essential for adequate protection 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 Davie and Price[39]). Vodock and Kent[112] address aspects of initial block usage insisting that they should be pseudo-random for CBC. The rationale given here and in various other texts is clearly wrong. For example Schneier states that an initial block can be a serial...
Clarke and Jacob [36] have shown that such an approach is potentially disastrous; the approach of their algorithm would allow a third party to create a 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 smaller than the block size (e.g. 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 content of a shift register is initialized with some value. The content of the shift register is encrypted as a block, and the leftmost byte of the result is XORed with the next plaintext byte to produce a ciphertext byte. The content of the shift register is now shifted left by 8 bits and the most centrally created ciphertext byte is placed in the rightmost byte of the shift register and the procedure is repeated. The decryption procedure is easily obtained.

The content of the register is now shifted left by 8 bits and the most centrally created ciphertext byte is placed in the rightmost byte of the register and the procedure is repeated. The decryption procedure is easily obtained.

\[ \text{XOR} \]

\[ \text{Cipher} \]

\[ \text{Feed} \]

\[ \text{back} \]

\[ \text{Key} \]

\[ \text{Left} \]

\[ \text{most} \]

\[ \text{byte} \]

\[ \text{Left} \]

\[ \text{most} \]

\[ \text{bit} \]

\[ \text{XOR} \]

\[ \text{Shift} \]

\[ \text{Register} \]

\[ \text{Encrypt} \]

\[ \text{CFB} \]

\[ \text{OFB} \]

\[ \text{Output} \]

\[ \text{Feedback} \]

\[ \text{Mode} \]
Schneier states that the initial vectors for CFB and OFB should be different for each message encrypted, although there is no additional benefit from sending them encrypted [99]. Veydec and Kenney 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 Davie and Price [39] for details.

Other modes are possible, e.g., Counter mode (like OFB but with the contents of the registers simply incremented each time, i.e., no feedback), Block Chaining mode (where the input to the encryption is the XOR of all previous cipher text 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 cipher text blocks). There are a variety of other modes which are somewhat exotic; we shall not describe them here.

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.
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 the cryptanalysis become easy). A keystream generator comprises a finite state machine and an output function. Figure 5 shows two basic approaches to stream cipher: output feedback mode (where the value of the key affects the next state) and the output function is pretty straightforward; and Counterc mode (where the key affects the output function and the next state is straightforward, typically a counter increment).

It is also possible to use block ciphers as keystream generators (e.g. use Counterc mode and select the leftmost bit of the encrypted block output).

For detail 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 public announcement on the topic was the classic paper by Whitfield Diffie and Martin Hellman in 1976 [44].

2.3

D M H 1976 44. I

13
Each principal \( A \) is associated with some key pair \((K_a, K_{a^{-1}})\). The public key \( K_a \) is made publicly available but the private key \( K_{a^{-1}} \) does not reveal all the principal \( A \) does not reveal all the principal's secrets. Any principal can encrypt a message \( M \) using \( K_a \) and only principal \( A \) can decrypt using \( K_{a^{-1}} \). 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 \( K_a \)) to a meaningful plaintext message \( P \), then it is assumed that the ciphertext \( C \) have been created by \( A \) using the key \( K_{a^{-1}} \). This can be used to guarantee the authenticity of the message. The most widely known public key algorithms that allow such use was developed by Rivest, Shamir and Adleman [92] and 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 Goldman [49]).

The RSA algorithm [92] works as follows:

1. pick two large primes \( p \) and \( q \), let \( n = p \times q \)
2. choose \( e \) relatively prime to \( \varphi(n) = (p - 1)(q - 1) \)
3. use Euclid's algorithm to generate a \( d \) such that \( e \times d \equiv 1 \pmod{\varphi(n)} \)
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 \( C = M^e \pmod{n} \).
6. the encrypted block \( C \) is decrypted by calculating \( M = C^d \pmod{n} \).

Encrypting and decrypting 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 \( K_{a^{-1}} \) and then encrypting the resulting \( B \)'s public key \( K_b \). \( B \) uses his private key to decrypt and then uses \( A \)'s public key to obtain the original message.

Public key algorithms stand to rely on the (supposed) computational difficulty of solving certain problems (e.g. factoring the Diophantine algorithm and factoring primes 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 common. Today's algorithms use larger key lengths of 1024 or even 2048 bits.
RSA were thought to be very secure; today 512 bits are considered a minimum requirement (and 1024 bits are recommended). Sheer processing capability also affects the ability of public key encryption. Public key algorithms are generally much slower than symmetric algorithms.

Schieber[99] gave 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]. Willelt 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 attack 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 cryptography[81].

2.4 One-way Hash Algorithms

We shall often require evidence that a message that has been sent has not been modified 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 the message and corresponding digest agree.

Hash functions are largely intended 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 $E(K_a : H(M))$ and sending the message $M$ with the newly calculated encrypted hash value. On receiving the message, $B$ can calculate $H(M)$ and then $E(K_a : H(M))$ and check whether the value agrees with the encrypted hash value received. Since the amount of encryption is small this is quite efficient means to demonstrate authenticity (assuming $A$ and $B$ do not take message from the other).
Notations

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

A protocol consists of a sequence of messages between principals and will be described using the standard notation. Principal are generally denoted by capitals such as A, B, and S (for a server). The sequence of messages $(1) \ A!B:\ M_1$ $(2) \ B!S:\ M_2$ $(3) \ S!B:\ M_3$ denote a protocol in which $A$ sends $M_1$ to $B$, then $B$ sends $M_2$ to $S$ who then sends $M_3$ to $B$. An attack on a protocol often involves one principal pretending to be another. We denote a malicious principal by $Z$. The notation $Z(A)$ denotes the principal $Z$ acting in the role of $A$.

A number generated by a principal $A$ is denoted by $N_a$. Such numbers are intended to be used only once for the purpose of the current run of the protocol and are generally termed nonces. We shall some time refine the notion of a nonce to include a time stamp and distinguish between sequence numbers and truly 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

$$(1) \ A\ B:\ A; E(K_{ab}: N_a)$$

denote that in the first message of the protocol $A$ sends to $B$ a message whose components are an identifier $A$ to $B$ with an encrypted nonce $E(K_{ab}: N_a)$. ISO (61)
In this section we provide an overview of various forms of authentication protocols in use today. At the highest level we have categorized them according to the principal cryptographic approach taken, i.e. symmetric key or public key. We also distinguish 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 message involved in the protocols (e.g. one-pass, two-pass, three-pass etc.) and whether one principal wishes to convince the other 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 identity 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] shown below. It consists of the single message:

\[
(1) \text{A} \xrightarrow{!} \text{B} : \text{Text} 2; E(K_a b) : [T_a ; N_a] \xrightarrow{} \text{B}; \text{Text} 1)
\]

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 \(T_a\) or a sequence number \(N_a\) depending on the capabilities of the environment and the communicating principals. On receiving this message, B, who believes that the key \(K_{ab}\) 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 an acceptable value. Note here that if an 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

\[
(1, 2) \text{A} \xrightarrow{!} \text{B} : 2 E( : B 1) \xrightarrow{} \text{H} ; (.) A ( . . ) - \xrightarrow{} ( . . ) . \text{T} \xrightarrow{} \text{O} . \text{A} , \text{A} , \text{A} . \text{N} \xrightarrow{} (1) \text{N} ( \cdot \text{B} \cdot \text{B} ) . \text{T} \xrightarrow{} \text{O} \cdot \text{A} \xrightarrow{} \text{A} \cdot \text{B} . \text{B} \cdot \text{F} 6 \xrightarrow{} \text{I} - . \text{I}
\]
The nonce should be random. If the nonce were an sequence number, or otherwise predictable, an 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 received was created at some time by B (but not necessarily in response to his most recent challenge).

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 checksum functions. Essentially, a message is sent to get 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 with a trusted third party (TTP) are by far the most numerous in the literature. The most celebrated protocol is the Needham-Schroeder Symmetric Key Authentication protocol [87].
In this protocol, A requests from the server S a key to communicate with B. He includes a random nonce Na generated specifically for this run of the protocol. This nonce will be used by A to ensure that message (2) is timely.

S creates a key Ka and creates message (2). Only A can decrypt this message successfully since he possesses the key Ka. Doing so, he will obtain the key Ka and check that the message contains the nonce Na.

A passes on to B the encrypted message component E(Ka : Ka : A).

Principal B decrypts this message to discover the key Ka 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 Ka, decrypts it, forms Nb, encrypts it, and sends the result back to B as message (5).

B decrypts this and checks that it is correct. The purpose of this exchange is to convince B that A is genuinely operational (and that message 3 was not imply 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 Ka 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 also 6.3.5), the Owtay-Rees Protocol [91] (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 Carlsten's secret key initiators 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 time stamps. The Denning Sacco Convention 19
Key Protocol replace the first three messages of the Neesham Schroeder protocol with:

1. A ! S: A ; B
2. S! A: E( Ks: Tb; B; Ksb; T; E( Ks: A ; Ksb; T ))
3. A ! B: E( Ks: A ; Ksb; 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 a window centered on the respective local clock time).

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

1. A ! S: A ; E( Ks: Ta; B; Ksb)
2. S! B: E( Ks: Ts; A; Ksb)

A is trusted to generate a session key Ksb. 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 (also trusted timestamping). Davidson and Swick provide more key translation services[40].

Some protocols allow keys to be used in more than one session. These are typically two-party protocols. The first party involves a principal A obtaining a 'ticket' for communication with a secondary principal B. The ticket generally contains a session key and is encrypted so that only the receiver B can decrypt it. In the second party of the protocol A presents the ticket to B when he wishes to communicate; he may do this several times (until the ticket expires). These are usually called repeated authentication protocols. Such protocols have been devised by Keene et al. [68] and also by Neumann and Stubben [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 wide spread 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 implementation)

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 wide spread 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 implementation...
The 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 paper [87]:

1. \( A \rightarrow S \): \( A \); \( B \)
2. \( S \rightarrow A \): \( E(K_s^{-1}, K_b; B) \)
3. \( A \rightarrow B \): \( E(K_b; N_a; A) \)
4. \( B \rightarrow S \): \( B; A \)
5. \( S \rightarrow B \): \( E(K_s^{-1}, K_a; A) \)
6. \( B \rightarrow A \): \( E(K_a; N_a; N_b) \)

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 requests signed under its own private key \( K_s^{-1} \).

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 \( N_a \) to get \( A \)'s identifier. \( B \) decrypts this to obtain the challenge, forms a challenge of his own \( N_b \) 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 \( N_b \) using \( B \)'s public key 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.

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

Public key cryptography may also be used to provide digital signatures. RSA [92] can be used to sign a message by encrypting under the 21
private key. It can also be used to sign a hash value of a complete message. The actual message can also be sent with the encrypted hash value appended. A major alternative to the use of RSA developed, amid controversy, by the United States National Security Agency (NSA) is the Digital Signature Algorithm. It is based on ElGamal 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 signature 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 (but seemingly very effective) is the 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 appears to tolerate fairly poor mechanisms of symmetric encryption.

3.5 Other Forms of Protocol
There are many other types of authentication protocols. 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 standards protocols can be found in [61], [62], [63], [64], [65]. Recent protocols include a beacon protocol by Berrey et al [66] and a robust password exchange protocol by Hausner 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 treaties [100]. Anderson provides an indication of how electronic payment systems work [9]. The same author discusses societal and legal aspects of cryptography [8], [7].
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 $K^b$ 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 Schneier demonstrated that the protocol was flawed [42]. Consider message (3). Although $B$ decrypts the 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 $S$ apart from the current protocol run. Thus, suppose a previously distributed key $K^0$ has been compromised (for example, by cryptanalysis) and is known to an intruder $Z$. $Z$ may have monitored the network when the corresponding protocol run was executed and recorded message (3) consisting of $E(K^0: K^0: A; B)$. He can now fool $B$ into accepting the key as new by the following protocol (omitting the first two messages):

(3) $Z(A) \rightarrow B : E(K^0: K^0: A; B)$

(4) $B \rightarrow Z : E(K^0: K^0: N : b)$

(5) $Z(A) \rightarrow B : E(K^0: K^0: N : b \oplus 1)$

$B$ believes he is following the correct protocol. $Z$ is able to form the correct response in (5) because he knows the compromised key $K^0$. He can now engage in communication with $B$ using the compromised key and masquerade as $A$. Denning and Schneier suggested that the problem could be fixed by the use of time stamps [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...
A type flaw arises when the recipient of a message accepts that message as valid but imposes a different interpretation on the bits sequence than the principal who created it.

For example, consider the Andrew Secure RPC Protocol (1)

\[
A \rightarrow B : \text{A} \rightarrow B : \text{E}(K_{ab}:N_0) \tag{1}
\]

\[
B \rightarrow A : \text{E}(K_{ab}:N_1;N_b) \tag{2}
\]

\[
A \rightarrow B : \text{E}(K_{ab}:N_b+1) \tag{3}
\]

\[
B \rightarrow A : \text{E}(K_{ab}:K_0;N_0) \tag{4}
\]

Here, principal A indicates to B that he wishes to communicate with him and sends an encrypted nonce \(E(K_{ab}:N_a)\) as a challenge in (1). B replies to the challenge and issues some how own by sending the message \(E(K_{ab}:N_a+1;N_b)\). A replies to B's challenge by forming and sending \(E(K_{ab}:N_b+1)\) to B. B now creates a session key \(K_0\) and distributes it (encrypted) together with a sequence number identifier \(N_0\) for future communication.

However, if the nonce and keys are both represented as bits 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:

\[
A \rightarrow B : \text{A} \rightarrow B : \text{E}(K_{ab}:N_a) \tag{1}
\]

\[
B \rightarrow A : \text{E}(K_{ab}:N_a+1;N_b) \tag{2}
\]

\[
A \rightarrow Z : \text{E}(K_{ab}:N_b+1) \tag{3}
\]

\[
Z \rightarrow B : \text{E}(K_{ab}:N_a+1;N_b) \tag{4}
\]

Thus principal A may be fooled into accepting the nonce value \(N_a+1\) as the new session key. The interpretations imposed on the plain text bits 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 freshness at attack, as pointed out by Burrows, Abadi and Needham [26].
The above protocol causes a key $K_{ab}$ created by the trusted server $S$ to be distributed to principals $A$ and $B$. The protocol is an identifier protocol.

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

(1) $A \rightarrow Z(B): M; A; B; E(K_a; N_a; M; A; B)$
(4) $Z(B) \rightarrow A: M; E(K_a; N_a; M; A; B)$

Here $A$ decrypts $E(K_a; N_a; M; A; B)$ checks for the presence of the nonce $N_a$ and accepts $(M, A, B)$ as the new key.

M, A, B
Since they were broadcasting 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 component of message (2) back to $B$. The attack is:

(1) $A!B: M$ $A; B; E(Ka:$ $N_a; M; A; B)$

(2) $B!Z(S): M; A; B; E(Ka:$ $N_a; M; A; B)$ $E(Kb:$ $N_b; M; A; B)$

(3) $Z(S)!B: M; E(Ka:$ $N_a; M; A; B)$ $E(Kb:$ $N_b; M; A; B)$

(4) $B!A: M; E(Ka:$ $N_a; M; A; B)$

He can now listen into conversation between $A$ and $B$ using the now publicly available key ($M; A; B$).

For the example soft flaw are given by Syversen [109] and Hwang et al [60].

4.3 Parallel Session Attacks

A parallel session attack occurs when two or more protocols 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:

(1) $A!B: E(Ka:$ $N_a)$

(2) $B!A: E(Ka:$ $N_a)$

Successful executions should convince $A$ that $B$ is a legitimate entity only $B$ could have formed the appropriate response to the challenge issued in message (1). In addition, the nonce $N_a$ may be used as a shared secret for the purpose of our other 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.

(1 1) $A!Z(B): E(Ka:$ $N_a)$

(2 1) $B!A: E(Ka:$ $N_a)$

(2 2) $A!Z(B): E(Ka:$ $N_a)$

(1 2) $B!A: E(Ka:$ $N_a)$

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 he expects to receive back in the

26
The replay attack is depicted as message (1.2). At the very least, believe that B sooperational. In fact, B may no longer exist. The attack is illustrated in Figure 8. Solid arrow signs indicate message from the first protocol, broken arrow signs indicate message from the second protocol.

\[
\begin{align*}
E(K_{a}:Na) \\
E(K_{a}:Na +1)
\end{align*}
\]

**Figure 8: Simple Parallel Session Attack**

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

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

\[
\begin{align*}
(1) A & : A E( K_{a}: Ta; B; K_{a}) \\
(2) S & ! B : E( K_{b}: Ts; A; K_{b})
\end{align*}
\]

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 key K_{b} and a time stamp T_{a} and forms message (1) which is sent to S.

On receiving message (1), S checks that the time stamp T_{a} is "timely" and, if so, forwards the key to B with its own time stamp T_{s}. B checks whether message (2) has a time stamp that is later than any other message it has received from S (and so will play off 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 no
timestamp of timeline is the same as \(B\) (i.e., accept message 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 \(B\).

\[
\begin{align*}
(1) \quad & A! S : A; E(K_a : T_{a}; B; K_{ab}) \\
(2) \quad & S! B : E(K_b : T_{b}; A; K_{ab})
\end{align*}
\]

\[
\begin{align*}
(10) \quad & Z(B) ! S : B; E(K_b : T_{0}; A; K_{ab}) \\
(20) \quad & S! Z(A) : E(K_a : T_{0}; B; K_{ab})
\end{align*}
\]

\[
\begin{align*}
(100) \quad & Z(A) ! S : A; E(K_a : T_{0}; B; K_{ab}) \\
(200) \quad & S! Z(B) : E(K_b : T_{0}; A; K_{ab})
\end{align*}
\]
The next section shows that 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 initialization data $R$, and the plaintext bits encrypted so far.

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

(4) $B$ ! $A$: $E(K_b A)$

(5) $A$ ! $B$: $E(K_b A)$

Suppose that the cipher stream forms message (4) is $b_1 b_2 \ldots b_n$. Now if $N_b$ is odd then the final plaintext bit (assumed to be the least significant bit) will be 1 and $N_b - 1$ will differ only in that final bit. On a bit-by-bit encryption basis, the cipher stream forms 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 50% 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 time stamp the then scope of the is seen to be greater (but seems unreported in the literature).

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

(3) $A$ ! $B$: $E(K_b A)$

Flipping the final bit of this message could turn $A$ into a $C$ under decryption. Since $A$ knows the key $K_b A$ 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 cipher text blocks we can recover the plaintext block corresponding to the second (provided we

29
Suppose that \( E(K : P_1 P_2 P_3) = C_1 C_2 C_3 \). Then \( C_1 C_2 C_3 \) looks like a cipher text that begins with initialization block \( C_1 \), and decrypts to \( P_2 P_3 \). Similarly \( C_1 C_2 \) decrypts to \( P_2 \) (uses \( C_1 \) as an initialization block) and \( C_2 C_3 \) decrypts to \( P_3 \). Thus we can see that without appropriate additional protection valid messages may be created if the contents are subsequences of generated messages. To distinguish this form of attack from those that follow we shall call all this form of flaw sequence flaw.

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

\[ S \quad \text{G} \quad \text{106} \]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Thus we can see that without appropriate additional protection valid messages may be created if the contents are subsequences of generated messages. To distinguish this form of attack from those that follow we shall call all this form of flaw sequence flaw.
Message Split

Corresponding 2K

Plaintext

Ciphertext

Figure 9: Splitting Attack on Cipher Block Chaining

\[ F \oplus (S \cdot A \cdot C \cdot B \cdot C) \]
ge ne r al l y be know n t o the par t i e sc om m uni c at i on pr ot ocol w ho m ay m i s us e t he i r know l e dge (see be l ow ). Of par ti cul ar not e ear e ini t i a t i on a t t ac ks— a t t ac ks that i nvo lv e mod u-
lat i on of the ini t i a t i on ve c t or $C_0$. Cons i der a c i phe r t e xt that st ar t s w i t h $C_0C_1$ and sup po se t hat w e know t h at the ini t i al pl ai nt e x t bl oc k w as $P_1$.

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

Now f or any de si r ed bl oc k val ue $W$ we have $W = W \oplus P_1 \oplus P_1$ s i nc e anyt hi ng XORe d w i t h i t s e l f i s 0. And s o w e have (subs t i t ut i ng f or t he se c ond $P_1$)

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

and s o $W = C_0 \oplus d_K(C_1)$

where $C_0 = W \oplus P_1 \oplus C_0$ and s o $C_0C_1 i s t he c i phe r t e xt c or re s pondi ng t o pl ai nt e xt $W$. In t hi s f as hi on w e c an r e pl ac e t he ini t i al kno w n pl ai nt e xt bl oc k $P_1 w i t h ou r o w n c h oi c e $W$. Thi s i s po t i ent i al l y ve r y di s t ur bi ng s i nc e t he re s to f t he m e s s age i s unaf f e c t e d.

As an e xam pl e of t he dange r oft hi s at t ac k, c ons i der a n othe r m e s s age (2) of t he Ne e dham Sc hr oe de r pr ot oc ol. W e c an r e c or d m e s s age (2) of a pr e vi ous run of t he pr ot oc ol be t wee n $A$ and $B$.

In par ti cul ar w e c an r e pl ay t he ol d m e s s age (2) af t e r m odi f yi ng t he ini t i al bl oc k f r o m t he ol d (and kno w n) val ue oft he non c e $N_a$ w i t h t he ne w one i s s ued i n t he cu r r e n tr un of t he pr ot oc ol . T h u s , w e c an i m pe r s onat e t he t r us t e d s e r ve r $S$.

Now c ons i der t he c ont e nt sofm e s s age (3) of t hat pr ot oc ol:

$$A \rightarrow B : E(K_{bS} : K_{aB} ; A)$$

S A (3), (3)

A

O

: B

(3)

N S T

(3)

(2)

(2)

(3) A B : E( : A)

32
We have illustrated various forms of cryptalgorithms depending on 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 protocol were given by Bellavin 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 key is paramount. Suppose your public key is \( K_y \) and an intruder's public key is \( K_i \). The intruder is able to decrypt any message encrypted with \( K_i \). Principals wishing to convey information to you secretly will encrypt what they believe is your public key. Thus, if the intruder can convince others that your public key is \( K_i \) then they will encrypt sensitive information using \( K_i \) and this will be readable by the intruder.

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

\[
\begin{align*}
(1) & \quad C \rightarrow AS: C \quad S: N_c \\
(2) & \quad AS \rightarrow C: AS; E(K_{as}^{-1}: AS; C; N_c; K_s)
\end{align*}
\]

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 \( N_c \) to ensure the freshness of the expected reply.

\( AS \) replies with message (2). The principal identifier \( AS \) issues a certificate to tell \( C \) which public key to use to decrypt the following cipher text. The components of the encrypted parts signify that the message was created by \( AS \), that his message has been created in response to a request from a client with nonce \( N_c \) and the requested public key is \( K_s \).

However, the reader may not recognize that 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:

\[
\begin{align*}
(1) & \quad Z_1: AS \rightarrow C: C \\
(2) & \quad Z_2: C \rightarrow AS: C; E(K_{as}^{-1}: AS; C; N_c; K_z)
\end{align*}
\]

33
<table>
<thead>
<tr>
<th>(1) C</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2) A</td>
<td>C</td>
<td>E( : A C )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCITT .509</td>
</tr>
<tr>
<td>B, A</td>
</tr>
</tbody>
</table>

4.6

<table>
<thead>
<tr>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>A. A</td>
</tr>
</tbody>
</table>

<p>| . A |</p>
<table>
<thead>
<tr>
<th>D</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>40:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(1) B</th>
<th>A</th>
<th>E( : A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2) A</td>
<td>C</td>
<td>E( : A ) B</td>
</tr>
<tr>
<td>(3) A</td>
<td>C</td>
<td>E( : B )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>. A</td>
</tr>
<tr>
<td>(3) B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>. T</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
</tr>
<tr>
<td>)</td>
</tr>
<tr>
<td>. H</td>
</tr>
<tr>
<td>,</td>
</tr>
<tr>
<td>C</td>
</tr>
</tbody>
</table>

| (3) |
| C. S | B |

| . C |
|? G |

<table>
<thead>
<tr>
<th>. I</th>
</tr>
</thead>
<tbody>
<tr>
<td>( CRC )</td>
</tr>
</tbody>
</table>

| . I CBC |

34
( ).

N . T 3

( ) 3 CBC -

. S

( ).

4.7

T -

. I -

M -

J M 85.

I -

. M B 80 -

. P K -

RSA 72.

G 51. P -

15, 54.

C 31.

. T ( ) CCITT .509

( ) BAN ) N 89

. I -

. C J -

CCITT .509 -

34. A N -

10, 11. A -

5.

35
4.8

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

However, there are 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 a considerable number of factors need to be taken into account. And one emphasises the management aspect 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.
In this section we review the major approaches to the specification and analysis of authentication protocols. Several methods have been tried, each with its strengths and weaknesses. We address them as follows:

- the use of existing formal methods to specify and analyze authentication protocols;
- the use of logical 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 mechanisms used. This is of course a strength since in producing a protocol specification we might not wish to specify a particular implementation mechanism.

However, it also highlights a gap in the formal support for protocol development: tools support for the identification of cryptographic system dependences.

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 (developers, evaluators, and Government agencies) is used to use formal specification and verification techniques for any aspect of security. Tools sets have been developed, or are being developed, and use can be made of the experience gained in other areas.

The first such attempt appears to have been that of Kemmerer, who in 1987 used the Ina tool 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, for example, to model, 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 proof are attempted. The Z language models the security

37
Language has been used in many areas of security outside of authentication. Within the UK, it is often the language of choice for specific government agencies. Details of these uses are omitted here. More recently, the notation has been used to specify authentication systems\[17\].

This method shows some promise as a tool to support merging.

The use of such state-based techniques seems limited. There is little or no attempt to model attacker (Kemmerer model as 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 know whether the specified operations actually do maintain security. Such methods are primarily concerned with the preservation of correctness rather than security.

Other forms of 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 provide analytical support for security in the face of an active intruder. Rubin and Honeyman\[95\] provide some details.

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

An authentic run of the protocol as specified (the protocol terminates successfully if the message sequence is what the protocol intended). The implementation of the protocol which comprises the various principals as agent must now be shown to satisfy the specification. The Failure Divergence Refinement (FDR) tool is used to check possible traces of the implementation against the specification. Roscoe and Gardiner have created a variety of heuristic techniques to prune the search space to make the model checking feasible.

The results have been very promising (subtle and other previously unknown protocol flaws have been discovered using the approach). For example, 17-year-old software flaw was found in the Needham Schröder Public Key protocol by Lowe using the FDR tool\[75\]. See also\[77\]. Roscoe and Gardiner provide an account of the initial results of the investigation\[94\]. The extension of the work to handle algebraic entities is also available\[47\][48]. A particularly pleasing part of the work is the willingness to...
5.2 The Use of Logics

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

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

Traditionally, trust issues have been dealt with using belief logics and issues of security have been dealt with using knowledge logics. Syverston [107] provides a good overview of how logics can be used for the analysis of authentication protocols. He indicate that it is possible to reason about both trust and security using either approach, but in practice he has found epistemic logics to be 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. These beliefs held by a principal change as she receives protocol messages. The authors provide an inference rule that defines how these belief states change. Thus, given an initial set of beliefs, the logic allows the analysis 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 the assumptions of a protocol and also what the goals are. In general, protocol descriptions have typically stated what the principals should do and not what they were trying to achieve.

The logic has stimulated a great deal of controversy. Nesset [89] provides an example of a clearly insecure protocol which is never less accepted by the BAN logic. Effectively, here hard keykey decryption

unde ra pri va te ke y and br oadc ast to t he ne t w ork. Si nc e t he c or re spond-
i ng pub li c ke y i s ge ne r al l y know n, t he m e s s age c an be de c orpi e t y by al l
t o obt a i n t he s e c r ets har e d ke y. In t he i r r e j o i nde r [27] t he BAN aut h ors
po i nt out t h e i r l ogi c de al t wi t h tr us t and notc onfi d en t i t y, st ati ng
that t he obvi ous pub li c at i on of t he s har e d ke y i n t he i ndi c at ed manne r
co nt r adi c t e d a be l i e fi n i t ess u i t abi l i t y f or us e.

Thi s w ou ld appe ar c or r e c t, but t he si tu at i on i s st i ll r at he rs at i sf yi ng.
Ad dit i o nal pr o b le m s have be e n i de nt i fi e d. Sne kke ne s [102] sh ow e d t hat
pe r mu ta t i o ns o f pr ot oc o ls t e p s l e f t t he r e s ul t s u n a f f e c t e d.

It i s p o s si bl e t h a t a pr i n c i p al m ay de c r ypt s o m e r a n d o m t e x t t o obta i n
s o m e p u t at i ve "f or m ul a" us i ng s o m e ke y t hat he ho l d s. For t he de c rypt-
t i o n t o s uc c e e d t he r es ul t s o f de c rypt i o n m us t be
m e a ni ngf ul i n s o m e w ay.

Gong, Ne e dham and Yahalom [55] in t roduc e t he not i on of r e c or gi s abi l i t y
i n t he i rl ogi c (ge ne r al r e fe r re d t o as GNY) t o c at e r for hi s. Al s o, t he or i gi-
nal BAN l ogi c as s um e s t h a t t he r e i s s u f fi ci e n t r e dundan c y i n a m e s s age
f or a pr i n c i p al t o de t e c t a m e s s age he hi m s e l f i m gi nat ed (t h u s re fe r-
t i on at t ac ks a re a s s um e d t o be c a t e r e d f or o u t s i de of BAN anal y s i s ). GNY l ogi c
m ake s or i gi nat i on e x pi li c i t. GNY al l ow s pr e c on di t i on s t o be at t ac he d t o
r ul e s t o ac hi e ve di f f e r e n t l e v e l s of be l i e f. Th us, di f f e r e n t l e v e l s of tr us t
ar e al l ow e d by t he l ogi c. Mos t BAN w or k c on c e nt r at e s on t he
a na l y s i s of pr ot oc o ls. W he n us e d f or de ve l opm e nt , pr o b le m s m ay ar i se be caus e c om-
pl e t e l y i nf e as i bl e pr ot oc o ls m ay be s pe c i fie d t hat ne ve r t he le s s a c hi e ve t he
d e s i r e d go al s ac c or di ng t o t he pr ot oc ol (e.g. by s pe c i fi ng t hat pr i nci p al s
s e nd m e s s age s t hat c ont ai n i nf or m at i on t he y s i m pl y d o nothave). Thi s
i s de al t wi t h by Gong [52] w hos e e xt e nde d l ogi c r e qui r e s t hat pr i nci p al s
m ake us e onl y of i nf or m at i on t hat i s s e ri o usl y a v a i abl e t o t he m.

Boyd and Mao [24] pr o vi de m any c r i t i c i s m s of BAN l ogi c (and ot he r
de s c e ndant s): t he f or m al i s at i on appr o ac h i s s om e w hatv ag u e; i tal l ow s
be l i e f s t hat m ay l e gi t i m at e l y be r e gar de d as nons e ns i c al (e.g. be l i e f i n a
nonc e) and t he m e t hod of de t er m i ni ng as s um pt i ons i s sa d hoc. Ins t e ad t he y
pr o vi de a l angu age f or de s c ri bi ng pr ot oc o ls and a pa r ti al l y m e c hani s e d
ap pr o ac h t o i de al i s at i on. As poi n t e d out b y Rub i n and Hone y m an [95]
the r e i s st i ll i nf or m al j udgem en tat w or k i n t he i de al i s at i on proc e s s. The
r e as oni ng pr oc e s s i s bac kw ar ds ( ra t he r t han f or w ar ds as i n BAN l ogi c),
t hus t he r e as oni ng pr oc e e ds f r om t he de s i r e d c onc l us i on t o de r i ve i ni t i al
be l i e f s.
tai n. Line ar pro gram mi ng me thods are use d to de term i ne the pre cise bound sof prob ab i l i t i e s.

K e s l e rand We del m odi f y BAN to al low the i nc or por at i on of pla int ext me ss age s. Thi s wi de nst he scope of what can be anal ys e d. The y al s o repl ac e the non-ve r i fi cat i on ru le of BAN wi t h a "has r e c e nt l y s ai d" ru le.

A r e c i pi ent of a me ss age no l onge r be l i e ve s t h e s ender of a me ss age be l i e ve s t he con tent s, r at he r he now j us t de du ce s t h e s ender s e nt i t r e c e nt l y. A ru le i s i nt r oduc e d to al low a pr i nc i pat to t ry ke ys t hat he has (or gen er at e) wi t hout ac t ual l y be l i e vi ng t hat he ke y i s app ropri at e f or t he me ss age i n que st i on. Ke s l e rand W e del ' s mos ti m por ants ug ge s t i on i s t he i nc or por at i on of a pas si ve e ave s dr oppe r i nt o t he sys t e m. By t he det er mi nat i on (by c l os ur e) of i nf or m at i on avai l abl e t o su ch an i nt r u de r, cer tai n type sof confide nt i t y br e ace he sc an be de t ec t e d (e.g. the Ne s-set tfl aw). The au thor spr ovi de an exam pl e of BAN' s i nabi l i ty t o de al wi t h a par all e ls e ses s i on at t ac k. Re c e nt w or k by Boyd and Mao has i ndi cat e d t hat ca re ne e dst o be t a ke n w he n c l e ar t ex ti som i t t e d butOor s c h ot di s put e s t he vi e w st he y t ake 

Ove r al l, BAN has pro ve d of subs t ant i al us e. I t o f t e n s e e ml i ke a mar ked i m pr ove m e n t on i t s suc c e s s ors w hi c h have adde d con ce pt ual app ar atus t o de al wi t h i t spe r c e i ve d de fi ci en c i es at the ex pe ri ence of cons id er abl e i nc re as e i n com pl e xi t y. Thi s i nde e d t he vi e w of R o ge r Ne ed ham (com m e nt i ng on GNY log i c). Ke s l e r and W e del not e t hat BAN e xt e ns i ons t e nd t o be e xt e ns i ons to t he or i gi nal BAN log i c , not t o i t s suc c e s s ors. BAN log i c has une ar t he d m any pro to col flaw sand pro vi de s a ve r y c os t -e f f e c t i ve me ans of de t ec t i ng (so me) flaw s. In t er m s of val ue f or mone y i t ha s m uc h t obe sa i d f or i t. The ru le w ou l d appe ar t o be "Tr y BAN fir s t; i t doe s n' t co s t a great de al and i t of t e n pr oduc e s r es ul ts." The me thod i s cl e ar l y not wi t h-out i t s di f fic ul t i es; i t shoul d be re gar de d as a us e ful t ool. BAN log i c de al s w i t h w i t h t he be l i e f sof tr us t w or t hy pr i nc i pal s; i t doe s not de al wi t h con fi-
indicates how various deficiencies of the standard notation can be overcome by providing rules for a standard protocol specification into a CKT5 logic specification. Snekkene has shown that the Brookman method is insufficient to detect multi-role flaws, i.e., where a principal does not restrict himself to playing just one role. He also suggests how to extend the Brookman approach to cope with the problem. Snekkene notes in his doctoral dissertation that principal operation is couched in rather complex formulae. Snekkene has also carried out significant work to use the HOL (Higher Order Logic) specification language and tools to specify and prove properties about protocols.

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 Millette al led to the development of the Interrogator tool. The user guide provides an updated account of the tool facilities. Protocols 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. 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 The 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. Snekkene [104] points out that the Interrogator does not allow the identification of guess-based attacks. BAN logic does not address the issue either. As far as we are aware the Interrogator has not discovered any new attacks. Meadow has developed an analysis tool based on term rewriting (the NRL Protocol Analyzer). The specification language is again prolog-based...
and fairly easy to follow. Principal sees beliefs and also knows 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 decrypting some plain text with the same key produces the 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]. Analysis processing is not entirely automated; lemmas for the tool to prove must be generated by the user.
6.1

6.1.1

ISO Symmetric Key One-Pass Unilateral Authentication Protocol

This protocol consists of a single message from one principal to a second. A secret key $K_{ab}$ is assumed to be shared between the two principals.

1. $A \rightarrow B$: Text 2; $E(K_{ab}: [T; A])$; $B \rightarrow A$: Text 1

The use of the text field is application specific. The choice between a sequence number $N_a$ and a timestamp $T_a$ 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 mean of challenge-response. The protocol is fairly familiar:

1. $B \rightarrow A$: $R_b$; Text 1
2. $A \rightarrow B$: Text 3; $E(K_{ab}: R_b; B)$

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

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 $B$.

1. $A \rightarrow B$: Text 2; $E(K_{ab}: [T; A])$; $B \rightarrow A$: Text 1
2. $B \rightarrow A$: Text 4; $E(K_{ab}: [T; B])$

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

H

(1) B  A :  1
(2) A  B :  3 E(  :  B  2)
(2) B  A :  5 E(  :  4)

O

(2) B  B

(1) O  (3) A

(1)  (2) .

6.1.5

I

(2) B

(1)  (2) .

6.1.6

T

(1) B  A :  B
(2) A  B :  A E(  : ( )  A )
(3) B  A :  B E(  : ( ) )

(4)

(1) A  B :  A E(  : )
(2) B  A :  E(  :  1 )
(3) A  B :  E(  :  1)
(4) B  A :  E(  :  )

A

T

(4)

A

A  ( . . )

B). T

45
6.2

**6.2.1 ISO One-Pass Unilateral Authentication with CCs**

\[ A \xrightarrow{K_a b} B; \text{Text} 1; f(K_a b(\text{Text} 1)) \]

**6.2.2 ISO Two-Pass Unilateral Authentication with CCs**

\[ B \xrightarrow{R_b} A; \text{Text} 1 \]

\[ A \xrightarrow{K_a b(R_b; B; \text{Text} 2)} \]

**6.2.3 ISO Two-Pass Mutual Authentication with CCs**

\[ A \xrightarrow{K_a b(\text{Text} 2; B; \text{Text} 1)} \]

\[ B \xrightarrow{T_b} A; \text{Text} 4 \]

\[ B \xrightarrow{K_a b(T_b; A; \text{Text} 3)} \]

**6.2.4 ISO Three-Pass Mutual Authentication with CCs**

\[ B \xrightarrow{R_b} A; \text{Text} 1 \]

\[ A \xrightarrow{R_a} B; \text{Text} 3 \]

\[ A \xrightarrow{K_a b(R_a; R_b; B; \text{Text} 2)} \]

\[ B \xrightarrow{K_a b(R_a; \text{Text} 4)} \]

**6.3 Symmetric Key Protocols Involving Trusted Third Parties**

**6.3.1 Needham Schröeder Protocol with Conventional Keys**

The original presentation is given in [87] (where 1 is protocol). More usual notations are adopted here.

\[ A \xrightarrow{B; A; \text{Na}} \]

\[ S \xrightarrow{E(K_s; \text{Na}; B; K_b; E(K_b; \text{Na}; A))} \]

\[ A \xrightarrow{E(K_b; \text{Na}; A)} \]

\[ B \xrightarrow{E(K_b; \text{Na}; \text{N}_1)} \]

\[ A \xrightarrow{E(K_b; \text{Na}; \text{N}_1)} \]

46
Them os tf am ous at t ac k is by De nni ng and Sac c o [42]. The r ei san ot he r pot e nt i al w e ak nes sw hi c h de pe nd on t he n at ur e of t he ass um pt ion s made about c rypt ogr aphi cs up por t . The m ai n pr o b l e m w i t h t hi spr ot oc oli st hat B has no w ay of e ns ur i ng t hat t he m e s s age (3) i s f r e s h. An i nt ru dre c an c om pr o mi se a ke y and t he n r e pl ay t he app or pri at e m e s s age (3) t o B and t he n c om pl et e t he pr ot oc ol . Al s o, A cou l d (s houl d he s o w i s h) al s o s poof m e s s age (3) i n t he s am e w ay (c aus i ng a st a l e ke y t o be r e ac c e pt e d by B).

If a s t r e am c i phe ri s us e d t hen t he di f fer en c e b et w e e n t he c i phe r te xt s i n (4) and (5) i s v e r y s m al l (one bi t ) and t hi s al l ows a s i m pl e at t ac k to be l aunc he d. The r e ade ri s r e f e r re d t o [21] . Se e al s o s e c t i on 4.4.1.

6.3.2 De nni ng Sac c o Pr ot oc ol

De nni ng and Sac c o s ugge s t e d f i xi ng t he f r e s hne s s f l aw i n t he Ne e dham -Sc hr oe de r pr ot oc ol a b o v e b y t he us e of t i m e s t am ps . The pr ot oc ol be c om e s :

(1) A ! S : A ; B
(2) S ! A : E( K a s : B ; K a b ; T ; E( K b s : A ; K a b ; T ))
(3) A ! B : E( K a s : B ; K a b )
T i sa t i m e s t am p .
B c an c he c k f ort i m e l i ne s so fm e s s age (3) (al l o w i ng f or c l oc k dr i f tand ne t w ork de l ays ). The r e i s no w no ne e d f or t he e xtr a e x-
-
6.3.3 O t w ay- Re e s Pr ot oc ol

The Ot w ay-Re e s Pr ot oc ol [91] i s a ll -k no w n pr ot oc olt hathas b e e n s h o w n t o be flaw e d. The not at i on of t he or i gi nali d f e r sf r om c om m on usage and s o t he f or m pr e s e nt e d he r e i s t hat gi ve n i n [26].

(1) A ! B : M ; A ; B ; E( K a s : N a ; M ; A ; B )
(2) B ! S : M ; A ; B ; E( K a s : N a ; M ; A ; B ) ; E( K b s : N b ; K a b )
(3) S ! B : M ; E( K a s : N a ; K a b ) ; E( K b s : N b ; K a b )
(4) B ! A : M ; E( K a s : N a ; K a b )

I n t he a b o v e M i san onc e (a r un i de nt i fie r ). In m e s s age (1) A s e nd st o B t he pl ai nt e xt M ; A ; B and an e nc rypt e d m e s s age r e adabl e on l y by t he s e r ve r S o f t he f or m s how n. B f or w ar ds t he m e s s age t o S t oge t he r w i t h a s i m i lar e nc rypt e d c om pone nt . The s e r ve r S de c rypt s t he m e s s age c om pone nt s and c he c k s t h at he c om pone nt s M ; A ; B ar e t he s am e i n bot h m e s s age s . If so, t he n it g e ne r at e sa ke y Ka b and s e nd s m e s s age (3) t o B w hi c h f or w ar ds p a r t o f t hem e s s a ge t o A. A and B w i l l u s e t he ke y Ka b on l y i ft hem e s s age
6.3.4


(1) A B : A
(2) B A : E( : A )
(3) A : A B E( : A )
(4) A : E( : B E( : A ))
(5) A B : E( : A )
(6) B A : E( : )
(7) A B : E( : 1)

T

B 21).

6.3.5


(1) A : A E( : B )
(2) B : E( : A )

A O (1)

B B (2)

. T .

48
The first way it can be attacked is by simply replaying the first time with within an appropriate time window - this will cause re-authentication since it 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 the event occurs.

(1) \( A \rightarrow S: A; E(K_a: T_0; B; K_{ab}) \)

(2) \( S \rightarrow B: E(K_b: A; N_a; N_b) \)

(3) \( S \rightarrow A: E(K_a: B; K_{ab}; N_a; N_b); E(K_b: A) \)

(4) \( A \rightarrow B: E(K_a: A; K_{ab}) \); \( E(N_a; N_b: N_b) \)

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 not noticed by Clark and Jacob (many protocols are equally susceptible).

(1) \( A \rightarrow B: A \)

(2) \( B \rightarrow E(K_a: A) \)

(3) \( A \rightarrow E(K_a: B) E(K_b: A) \)

(4) \( e \rightarrow A \rightarrow B: E(K_a: A) E(K_b: A) \)

The following protocol is a modified form of this protocol can be found in [108].
6.3.7 Carl sen’s Secret Key Initiator Protocol

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

(1) $A \rightarrow B : A$

(2) $B \rightarrow A : B$

(3) $B \rightarrow E( : A) E( : B)$

(4) $B \rightarrow A : E( : B) E( : )$

(5) $A \rightarrow B : E( : )$

6.3.8 ISO Four-Pass Authentication Protocol

(1) $A \rightarrow B : TV Pa; B; Text$

(2) $S \rightarrow A : Text4; E(Ka s: TV Pa; Kb s; Text3)$

(3) $A \rightarrow B : Text6; E(Kb s; [Ts j N s]; Ka b; A; Text2)$

(4) $B \rightarrow A : Text8; E(Ka b; [Tb j N b]; A; Text7)$

6.3.9 ISO Five-Pass Authentication Protocol

(1) $A \rightarrow B : Ra; Text1$

(2) $B \rightarrow S : Rb j A; Text2$

(3) $S \rightarrow B : Text5; E(R0 b; Ka b; A; Text4)$

(4) $B \rightarrow A : Text7; E(Ra; Ka b; B; Text3)$

(5) $A \rightarrow B : Text9; E(Ka b; Ra; Rb; Text8)$

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 when ever a response finishes, the initiator of the protocol is in fact the principal claimed in the initiator message.

Woo and Lam start with a protocol $\Pi$ 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 principals can detect the replay of message they have created.

50
The protocol $\Pi_1$.

(1) $A \rightarrow B : A$
(2) $B \rightarrow A : $
(3) $A \rightarrow B : E( : A B )$
(4) $B \rightarrow E( : A B E( : A B ))$
(5) $B \rightarrow E( : A B )$

The protocol $\Pi_2$.

(1) $A \rightarrow B : A$
(2) $B \rightarrow A : $
(3) $A \rightarrow B : E( : A B )$
(4) $B \rightarrow E( : A B E( : A B ))$
(5) $B \rightarrow E( : A B )$

The protocol $\Pi_3$.

(1) $A \rightarrow B : A$
(2) $B \rightarrow A : $
(3) $A \rightarrow B : E( : A )$
(4) $B \rightarrow E( : A E( : A ))$
(5) $B \rightarrow E( : A )$

The protocol $\Pi_4$.

(1) $A \rightarrow B : A$
(2) $B \rightarrow A : $
(3) $A \rightarrow B : E( : )$
(4) $B \rightarrow E( : A E( : ))$
(5) $B \rightarrow E( : )$
The protocol \( \Pi \) can be attacked as follows:

1. \( Z(A) ! B : A \)
2. \( B(A) : \)
3. \( A \)
4. \( B(\ ) : E( : A ) \)
5. \( B( ) : B \)

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:

1. \( Z(A) ! B : A \)
2. \( B(Z(A)) : N_a \)
3. \( Z(A) ! B : G \)
4. \( B(S : E(K_b s : A ; G)) \)
5. \( Z(R) : B \)

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

1. \( Z(A) ! B : A \)
2. \( B(Z(A)) : N_b \)
3. \( Z(A) ! B : E(K_z s : N_a) \)
4. \( B(S : E(K_b s : Z; E(K_z s : N_a))) \)
5. \( Z(S) ! B : E(K_b s : N_a) \)

Similar attacks may be mounted against \( \Pi_1 \) and \( \Pi_2 \) etc. as stated above.

Woo and Lam assume explicitly that principals can detect some of the states they have created. Even if this were so (and we would prefer the mechanism to be part of the protocol), a pointer raised also by Low [76].

T : 117. H , , .

F , ,

1. \( (A) B : A \)
2. \( B(A) : \)
3. \( A \)
4. \( B(\ ) : E( : A ) \)
5. \( B( ) : E( : ) \)

S

L . E ( , L 76 )

52
Security of the protocol would still depend on the properties of the cryptosystem used. Thus, Woo and Lam note that $\Pi_i$ is not acceptable 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:

1. $B \rightarrow Z : B$
2. $(1) \rightarrow \{ A \rightarrow B : E(K_{zs} : P, Q, N_1, N_2) \rightarrow Z \}$
3. $(2) \rightarrow \{ S \rightarrow B : E(K_{zs} : P, Q, N_1, N_2) \rightarrow E(K_{qs} : P, Q, N_1, N_2) \}$
4. $(3) \rightarrow \{ P \rightarrow Q : E(K_{ps} : P, Q, N_1, N_2) \rightarrow E(K_{q} : P, Q, N_1, N_2) \}$
5. $(4) \rightarrow \{ Q \rightarrow S : E(K_{ps} : P, Q, N_1, N_2) \rightarrow E(K_{q} : P, Q, N_1, N_2) \}$

6.3.11

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

1. $P \rightarrow Q : P$
2. $(1) \rightarrow A \rightarrow B : A$
3. $(2) \rightarrow B \rightarrow E(K_{ps} : P, Q, N_1, N_2)$
4. $(3) \rightarrow E(K_{qs} : P, Q, N_1, N_2) \rightarrow E(K_{ps} : P, Q, N_1, N_2) \rightarrow S$
5. $(4) \rightarrow S \rightarrow Q : E(K_{ps} : Q, N_1, N_2) \rightarrow E(K_{ps} : Q, N_1, N_2) \rightarrow B$
6. $(5) \rightarrow B \rightarrow E(K_{ps} : Q, N_1, N_2) \rightarrow E(K_{ps} : Q, N_1, N_2) \rightarrow P$

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 a previously issued key. The attack consists of:

1. $P \rightarrow Q \rightarrow S \rightarrow P$
2. $(1) \rightarrow P \rightarrow Q : E(K_{ps} : P, Q, N_1, N_2) \rightarrow E(K_{ps} : P, Q, N_1, N_2) \rightarrow Q$
3. $(2) \rightarrow Q \rightarrow S : E(K_{ps} : P, Q, N_1, N_2) \rightarrow E(K_{ps} : P, Q, N_1, N_2) \rightarrow P$

The table below shows the results of the attack:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$P \rightarrow Q$</td>
</tr>
<tr>
<td>2</td>
<td>$Q \rightarrow { A \rightarrow B : A }$</td>
</tr>
<tr>
<td>3</td>
<td>$B \rightarrow E(K_{ps} : P, Q, N_1, N_2)$</td>
</tr>
<tr>
<td>4</td>
<td>$E(K_{qs} : P, Q, N_1, N_2) \rightarrow E(K_{ps} : P, Q, N_1, N_2) \rightarrow S$</td>
</tr>
<tr>
<td>5</td>
<td>$S \rightarrow { P \rightarrow Q : E(K_{ps} : P, Q, N_1, N_2) \rightarrow E(K_{ps} : P, Q, N_1, N_2) \rightarrow Q }$</td>
</tr>
<tr>
<td>6</td>
<td>$Q \rightarrow S : E(K_{ps} : Q, N_1, N_2) \rightarrow E(K_{ps} : Q, N_1, N_2) \rightarrow B$</td>
</tr>
<tr>
<td>7</td>
<td>$B \rightarrow E(K_{ps} : Q, N_1, N_2) \rightarrow E(K_{ps} : Q, N_1, N_2) \rightarrow P$</td>
</tr>
</tbody>
</table>

The protocol is vulnerable to this attack.
The following steps:

1. Back at these messages (with order reversed) to cause authenticity guarantee that its should.
2. Strong attack but indicates clearly that the protocol does not provide the accept the key. This is recorded in [37]. The above is not a particular communication intended for complete as normal. The second protocol proceeds with use that the response in the first protocol. The first protocol theなんてある property.

First the form such characteristic value (digest) principal.

This protocol comes from the classic paper [87] (where it is written). The low has recently found a more vicious attack based on

\begin{align*}
(1) & : 1 \\
(2) & : 1 \\
(3) & : 2 \\
(4) & : 2 \\
(5) & : E( : 1 2) E( : 1 2) \\
(6) & : E( : 1 2) E( : 1 2) \\
(7) & : E( : 2 )
\end{align*}

6.4.1 Needham-Schroeder Signature Protocol

6.4 Signature with Conventional Key Encryption

Note want to send a message to A

\begin{align*}
& P A & B \\
& F & ( ) & C & ( - ) \\
& H & (1) \\
& C & ( ) & A & N \\
& B & A & H & & 54
\end{align*}
6.5 Symmetric Key Repeated Authentication protocols

6.5.1 Kerberos Version 5

The protocol in three parts each of which is 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 server 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 scripts of the principals who share (or will share) them. Thus, \( K_{cg} \) denotes the key for secure communication between \( C \) and \( G \). We use \( K_u \) to denote the key used to encrypt communications between \( A \) and \( C \) on behalf of the user \( U \). It is a key obtained by having \( U \) store his password. \( A \) stores this password, \( C \) will request it.

The first part of the protocol is only concerned with \( C \) and \( A \).

(1) \( C \) ! \( A \): \( U \); \( G \); \( L_1 \); \( N_1 \)

(2) \( A \) ! \( C \): \( U \); \( T_{cg} \); \( E( : A C ) \)

where \( T_{cg} = E( : A C ) \) \( \langle \text{ticket} \rangle \).

In message (1) the client \( C \) informs the key distribution centre \( A \) that he wishes to communicate on behalf of user \( U \) with ticket granting server \( G \). A lifetime \( L \) and a nonce \( N_1 \) are sent too. \( A \) generates a new key \( K_{cg} \) for this purpose and encrypts it under key \( K_u \) its shares (or will share when \( U \) enters his password) with \( C \). It also forms a 'ticket'.
Figure 10: Kerberos Exchange

That contains the user identity, the client identity, the identity of G, the new key K_{CG} togethe with the timestamp information. The ticket lifetime is the interval over which the ticket is considered as valid. This ticket is encrypted using the key K_{AG} shared between A and G.

C uses the key K_{UC} to decrypt the third component of the message (2) and obtain the key K_{CG} which it can now use to communicate with G. This is carried out in the second part of the protocol described below:

(3) C! G : S; L_{2}; N_{2}; T_{CG}; A_{CG} (4) G! C : U; T_{CS}; E(K_{CG}: S; K_{CS}; T_{0}start; T_{0}expire; N_{2})

Where A_{CG} = E(K_{CG}: C; T_{0}); T_{CS} = E(K_{CG}: U; C; S; K_{CS}; T_{0}start; T_{0}expire)

The result of the above is that G issues C with a ticket T_{CS} and a key K_{CS} to communicate with S. The authenticator A_{CG} ensures timestamps of message (3).

In the third part of the protocol C uses the newly obtained key K_{CS} and ticket T_{CS} to obtain the services of S.

(5) C! S : T_{CS}; A_{CS} (6) S! C : E(K_{CS}: T_{0})
Ac = E(Kcs : C ; T0)

Hef or ms the raut he nt i c at or Ac, and s e nds her e s ul tt o S to ge t he rw i t h
the ne w l y ac qui r e d e nc ryp t e d t i c ke tasm e ss age (5).

Sc ar r i e s out de c r yp-
ti on on t he t i c ke tt o obt ai n t he se s s i on ke y Kcs and t he n us e s t hi s ke y t o
obt ai n t he aut he nt i c at i on i nf or m at i on. Ife ve r yt hi ng i s i n orde r , m e ss age
(6) i s r e t ur ne d.

6.5.2 Neum an St u bbl ebi ne

Thi spr ot oc olc ont ai nst w o par t s : one t o br i ng about t he e xc hange of som e
ti c ke tand t he s e con d i s a pr ot oc ol f or m ul t i pl e aut he nt i c at i ons. W e c al l
t he s e pr ot oc ol s r e pe a t e da ut he nt i c at i o n pr o t o c o l s. In t he Ne um an St u bbl e bi ne
Pr ot oc ol[ 90] gi ve n be l ow , t he fir s tf our m e ss age sar e t he i ni t i al pr ot oc ol.

(1) A ! B : A
(2) B ! S : B E(Kbs : A ; N a ; t b ) ; N b
(3) S ! A : E(Kas : B ; N a ; Ka b ; t b ) ; E(Kbs : A ; Ka b ; t b ) ; N b
(4) A ! B : E(Kbs : A ; Ka b ; t b ) ; E(Kas : N b )

At t ac ks have be e n s uc c e s s f ul l y m ount e d on bot h par t s of t he pr ot oc ol.
Pos si bl e at t ac ks c an be f ound i n [ 60] . The fir s tat t ac k on t he i ni t i al pr o-
toc ol si s gi ve n be l ow .

(10) Z(A) ! B : A
(20) B ! Z(S) : B E(Kbs : A ; N a ; t b ) ; N b
(30) Z(A) ! B : E(Kas : B ; N a ; Ka b ; t b ) ; E(N a : N b )

The s ubs e que nt pr ot oc ol c an t he n be at t ac ke d as f ol l ow s :

(10) Z(A) ! B : E(Kas : B ; N a ; Ka b ; t b ) ; E(N a : N b )
(20) B ! Z(A) : N 0 b ; E(Ka b : N 0 a )
(30) Z(A) ! B : E(Ka b : N 0 b )

A

P 90


(1) A B : A
(2) B : B E(A : A )
(3) A : E(B : B ) E(A : A )

T

(1) A B : E(A : A )
(2) B A : E( : )
(3) A B : E( : )

A

P 60 . T

(1) (A) B : A
(2) B ( ) : B E( : A )
(3)
(4) (A) B : E(A ( ) ) E( : )

T

(1) (A) B : E(A ( ) )
(2) B (A) : E( : )
(3) (A) B : E( : )
The following parallels session can be used:

(1) \( Z(A) : N_0 \); \( E(K_{bs}:A;K_{ab};t_{b}) \)

(2) \( B : Z(A) : N_0 \); \( E(K_{ab}:N_0;A) \)

In this attack the initialization key is recorded from a previous legitimate run.

6.5.3 Kehne Langendorf Schonw alder

Here is the KLS repeated authentication protocol. The first five messages form the ticket distribution part. The key \( K_{bs} \) is known only to \( B \).

(1) \( A : B \); \( N_0 \)

(2) \( B : Z(A) : N_0 \); \( E(K_{ab}:N_0;A) \)

The repeated protocols:

(1) \( A : B \); \( E(K_{bs}:t_{b};A;K_{ab}) \)

(2) \( B : A \); \( N_0 \); \( E(K_{ab}:N_0;A) \)

(3) \( A : B \); \( E(K_{ab}:N_0) \)

(4) \( B : A \); \( E(K_{ab}:N_0;A;K_{ab}) \); \( E(K_{bs}:t_{b};A;K_{ab}) \); \( E(N_c) \)

(5) \( A : B \); \( E(K_{ab}:N_c) \)

There are several authentication parts of the protocol subject to an attack that is identical to the parallel session attack on the Neumann-Stubbins protocol.

6.5.4 The Kao Chow Repeat An authentication protocol

In 1995, Kao and Chow proposed a similar repeated authentication protocol that was not acceptable to the attacks on the Neumann-Stubbins protocol [67].

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

(2) \( B : A \); \( E(K_{bs}:A;B;N_0;K_{ab}) \); \( E(K_{ab}:N_0;A;K_{ab}) \)

(3) \( B : A \); \( E(K_{bs}:A;B;N_0;K_{ab}) \); \( E(K_{ab}:N_0;A;K_{ab}) \)

(4) \( A : B \); \( E(K_{bs}:t_{b};A;K_{ab}) \); \( E(K_{ab}:N_c) \)
This protocols offers when a session key is compromised (as in the Denning Sacco attack on the Needham Schroe der protocol). The author for proposes (in the same paper) to use a different key purely for the handshake. The protocol now becomes:

\begin{align*}
(1) & \ A \rightarrow \ S \ : \ A, B, Na \\
(2) & \ S \rightarrow \ B \ : \ E(\ K_a \ : \ A, B, Na \ ) \ E(\ : \ A, B \ ) \\
(3) & \ B \rightarrow \ A \ : \ E(\ K_b \ : \ A, B, Na, Kb, Kt ) \ E(\ : \ Kb, Kt ) \\
(4) & \ A \rightarrow \ B \ : \ E(\ : \ ) \ E(\ Kb \ : \ A, B, Ta, Kb, Kt )
\end{align*}

\section*{6.6}


\subsection*{6.6.1}

\begin{align*}
(1) & \ A \rightarrow \ B \ : \ C, B \ 2 \ E(\ : \ B \ ) \ 1
\end{align*}

\subsection*{6.6.2}

\begin{align*}
(1) & \ B \rightarrow \ A \ : \ 1 \\
(2) & \ A \rightarrow \ B \ : \ C \ B \ 3 \ E(\ : \ B \ ) \ 2
\end{align*}

\subsection*{6.6.3}

\begin{align*}
(1) & \ A \rightarrow \ B \ : \ C \ B \ 2 \ E(\ : \ B \ ) \ 1 \\
(2) & \ B \rightarrow \ A \ : \ C \ A \ 4 \ E(\ : \ A \ ) \ 3
\end{align*}

T
6.6.4 -

(1) B A : 1
(2) A B : C B 3 E( : B 2)
(3) A B : C A 5 E( : A 4)

T - (3).

6.6.5

(1) A B : C 1
(1) B A : C 2
(2) B A : A 6 E( : A 5)
(2) A B : B 4 E( : B 3)

6.6.6

(1) B A : B E( : B)
(2) A B : E( : ( ) A )
(3) B A : E( : ( ))

6.6.7

I D -H

A B. L . T

(1) A B :
(2) B A :

A B

(1). B
A (2). A B
. T . T . T

60
6.7

6.7.1

Needham–Schroeder Public Key Protocol

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

\[\begin{align*}
(1) & \quad A \rightarrow B \\
(2) & \quad A \rightarrow E(\quad B) \\
(3) & \quad A_B \rightarrow E(\quad A) \\
(4) & \quad B \rightarrow B A \\
(5) & \quad B \rightarrow E(\quad A) \\
(6) & \quad B \rightarrow E(\quad ) \\
(7) & \quad A \rightarrow E(\quad ) \\
\end{align*}\]

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:

\[\begin{align*}
(3) & \quad A \rightarrow Z \\
(4) & \quad Z \rightarrow B \\
(5) & \quad B \rightarrow Z \\
\end{align*}\]

6.8

SPLICE/ASA 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 time stamp and L is a lifetime.

\[\begin{align*}
(1) & \quad C \rightarrow AS \\
(2) & \quad AS \rightarrow C \\
(3) & \quad C \rightarrow S \\
(4) & \quad S \rightarrow AS \\
(5) & \quad AS \rightarrow S \\
(6) & \quad S \rightarrow C \\
(7) & \quad S \rightarrow C \\
\end{align*}\]

This protocol has been shown to be flawed (in different ways) by Hwang and Chen [59] and also Gavin Lowe.
In the second attack it is possible to impersonate the server:

modified SPLICE/AS protocol has recently been shown by Clark and Jaffar (the flaws that they had identified) in the protocol presented above. This

Hwang and Chen [59] proposed an enhanced protocol to overcome the

6.8.1 Hwang and Chen’s Modified SPLICE/AS

possibility of time window to achieve authentication.

In the third attack (by Gavin Lowe) message (3) is replayed within the

possessed.

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

For the purpose of the attack we need only consider messages (3) and

(2)

(1)

(4)

(3)

(5)

(6)

AS

C

(6)

(1)

C

A

E( : A

(4)

( )

A

: A

E( : A

(5)

A

: A

E( : A

(6)

( C)

: C

E( : 1

I

( G L )

(3)

.

6.8.1

/ T

SPLICE

. T

SPLICE/AS

C

J

.

C

59

.

( )

SPLICE

.

(1)

C

A

: C

(2)

A

C

: A

E( : A

(3)

C

: C

E( : C

(4)

A

: C

(5)

A

: A

E( : A

(6)

C

: C

E( : 1

F

(3)

.

(3)

C

( )

: C

E( : C

(3)

: E( : E( : ))

(6)

: E( : 1

(6)

( )

C

: C

E( : 1

62
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 intended to provide a means for a conventional communication key Ka to be generated by A and passed to B securely. S provides certificates to A and B.

(1) $A \rightarrow S$: $A$;
(2) $S \rightarrow A$: Cert$_A$; Cert$_B$;
(3) $A \rightarrow B$: Cert$_A$; Cert$_B$; $E(K_b$: $E(K_{a^{-1}}$ a: $K_{ab}$; $T_a)$)

where Cert$_A$ = $E(K_{a^{-1}}$ s: $A$; $K_a$; $T_a)$ 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 time stamps signed by A and form a message of the form

(3) $B(A) \rightarrow C$: Cert$_A$; Cert$_C$; $E(K_c$: $E(K_{b^{-1}}$ a: $K_{ab}$))

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 of message1, message2, and 3 below. It has been shown to be flawed.

(1) $A \rightarrow B$: $A$; $E(K_{a^{-1}}$ a: $T_a$; $N_a$; $B$; $X_a$; $E(K_b$: $Y_a$))
(2) $B \rightarrow A$: $B$; $E(K_{b^{-1}}$ b: $T_b$; $N_b$; $A$; $N_a$; $X_b$; $E(K_a$: $Y_b$))
(3) $A \rightarrow B$: $A$; $E(K_{a^{-1}}$ a: $N_b$)

Attacks have been found by L'Ans on and Mitchell[12] and by the Burers 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 content of that component.

63
6.10

6.10.1 Shamir Rives

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 is assumed that encryption is commutative. It is known to be subject to a variety of attacks.

\[(1) A \rightarrow B: E(K_a: M) \]
\[(2) B \rightarrow A: E(K_b: E(K_a: M)) \]
\[(3) A \rightarrow B: E(K_b: M) \]

The first attack simply uses A as an oracle.

\[(1) A \rightarrow Z: E(K_a: M) \]
\[(2) Z \rightarrow B: E(K_a: M) \]
\[(3) A \rightarrow Z: M \]

Carl suggests that it might be possible to simply check whether the message returned in (2) is in fact encrypted, but it 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:

\[(1) A \rightarrow Z: E(K_a: M) \]
\[(2) Z \rightarrow A: E(K_a: M) \]
\[(3) A \rightarrow Z: M \]

6.10.2 Gong Mutual Authentication Protocol

This protocol 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. \(N_a, N_b\) and \(N_s\) are nonces.

\[(1) A \rightarrow B: A; B; N_a \]
\[(2) B \rightarrow S: A; B; N_a; N_b \]
\[(3) S \rightarrow B: N_s; \overline{f(N_s; N_b; A; P_b)} \]
\[(4) B \rightarrow A: N_s; h_b \]
\[(5) A \rightarrow B: h_a \]

..
In message (3) \((k; h_a; h_b)\) 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 \(f(N_s; N_b; A; P_b)\) to retrieve \((k; h_a; h_b)\) from the second item of the message. Also computes \(g(k; h_a; h_b; P_b)\) to check against the third item that tampering has not occurred. After receiving message (4), \(A\) computes \(f(N_s; N_a; B; P_a)\) to get \((k; h_a; h_b)\). If the value of \(h_b\) matches the one sent by \(B\) then \(A\) replays with message (5). The literature surveyed has not indicated 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:

(1) \(A!B: E(P:K_a)\)
(2) \(B!A: E(P:E(K_a:R))\)
(3) \(A!B: E(R:N_a)\)
(4) \(B!A: E(R:N_a;N_b)\)
(5) \(A!B: E(R:N_b)\)

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)

(1 1) \(A\) \((B) : E( : )\)
(2 1) \((B) A : E( : )\)
(2 2) \(A\) \((B) : E(E( : ))\)
(1 2) \((B) A : E(E( : ))\)
(1 3) \(A\) \((B) : E( : )\)
(2 3) \((B) A : E( : )\)
(2 4) \(A\) \((B) : E( : )\)
(1 4) \((B) A : E( : )\)
(1 5) \(A\) \((B) : E( : )\)
(2 5) \((B) A : E( : )\)
6.10.4

Daviss Wicke Privat e K ey Cer t ificates

The first protocol by Davi sand Sw i c k [40] is for ke y trans l at i on via a tr ust ed trans l at or T. The pro to col is gi ven be low:

(1) $B!A : E( Kb t : A ; ms g )$

(2) $A!T : E( Kb t : A ; ms g ) ; E( Kt : Kb t ; B ; Lb ) ; E( Kt : Ka t ; A ; La )$

(3) $T!A : E( Ka t : ms g ; B )$

On recei vi ng me s sage (3) A as sumes that ms g or i gin ated with B and was de stine d for A. If B ar rage sf or ms g = CX for some ident i fier C the n me s sage (3) be comes $E( C ; X B : Ka t )$. B can now use t hi st o m as que r ade as A in me s sage (1). Suf fic i e ntr e du ndancy w oul d ne e d to be placed i n the me s sage to pr e ve nt t hi s at t ac k (not i ced by Cli rk and Jac ob). The pro to col do e snote t i me l i ne s s.

A s c al ed up ve r s i on of the ke y trans l at i on s e r vi c e i sal s o pr e s ent e d.

(1) $B!A : E( Kb t : A )$

(2) $A!T : E( Kb t : A ) ; E( Kt : Kb t ; B ) ; E( Kt : Ka t ; A )$

(3) $A : E( Kb t : B )$

H

E( : A ) A

T

(1) $B : E( : B )$

(2) $B : E( : B ) ; E( : )$

A

(1) $A : E( : A ) ; B$

(2) $A : E( : A ) E( : B )$

(3) $A B : E( : A ) E( : A )$
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 Zakuidin, 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.
Reference


This paper sets out several heuristic rules which enable people to write good protocols, though the report says that they are neither sufficient nor 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 or not to attack" list for the designer.

There are two overarching principles.

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

Principal 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 format. Principle 2 seems clear, and helps the designer sort the individual protocols.

The remaining principles are:

Principal 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 (that attack on the Denning and Schachman Key Exchange protocol and the Wong and Lam protocol for checking the existence of a principle).

Principal 4
Be clear about why encryption is being done. Encryption is not cheap and not doing so exactly 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.

Principal 5
When a principal signs material that has already been encrypted it should not be inferred that the principal knows the content.
of the message. On the other hand, it is property to infer that the participants signs a message and the encryption for privacy knowns the content of that message.

Failure 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 success may not do for ensur-ing associations, 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 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 time stamps 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 allowed age of a message deduced to be valid. Further more the time maintenance mechanism everywhere become 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.


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

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 systems views need to be taken and that current system holders do not promote this. The goals of ATM system needs 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 irrelevant to real concerns. Furthermore, it would appear that in practice the military sector is prone to system failures. Anderson maintains that the TCSEC and ITSEC approaches are currently inappropriate for real-world needs.

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 processes of creating security systems.
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-world applications of cryptography (ranging from ATMs and utility tokenst to lottery ticketing terminals and post office franking machines). There are quite a few juicy examples of cryptographic failure 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 like cryptography, that the use of wire taps varies enormously between countries, that maintaining wire taps in digital fashion is very expensive and that there are actually attacks on individuals arise from unauthorized access to data and has little to do with listening in encrypted communications.

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 whole system 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 proving freshness and any assumptions being made).


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

1. 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 encryption);
2. There is a problem arising because the first message encrypts before signing which allows someone to 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);
3. 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 timestamp option is optional);
4. The working of the conditions for using RSA are incorrect (or rather the reasoning is wrong) and recommended hash function has been shown to be defective.

A good read. Further more, 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.


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].
In this paper the authors describe several weaknesses of the current Kerberos authentication system. The authors address potential attacks, secure time service provision, password guessing attacks, login spoofing, chosen plaintext attacks interalia. Solutions are suggested and assessed. Some interesting remarks are made concerning susceptibility to cipher texts explaining replay attacks. A very good read.

Thi spaper 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 occurs 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 (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 ElGamal.

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 properly are given.
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 secure as the cipher block chaining encryption that supports it. The protocol looks secure and compact.


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 simple weaknesses that have now become prudent practice. It is an informal development.

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

A good, simple and informative paper. The theme is that cryptographic protocol specifications do not state precisely the assumptions they make about the underlying cryptographic algorithms. Two very well-known protocols are given (Needham-Schroeder [88] and Owtewy-Rees [91]) and are shown to depend crucially on the underlying cryptographic algorithms. The danger of using Cipher Block Chaining (CBC) mode for DES (and others) is shown to allow various attacks (which depend on the implementation). An entrance and very simple example is given of how use of stream cipher for the use of challenge response can lead to an attack (both challenge and response are encrypted using the same key). The author provides new descriptions of what properties are required of cryptographic algorithms and concludes that nonce should not be encrypted in more than one message using the same key.

Problems with CBC mode of encryption are in fact what this paper indicates. The paper is well-written and highlights a much neglected area.
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 function. The principals exchange nonce 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 more 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 so interesting that it presents a plausible (but faulty) implementation of the intent of the protocol. The author concludes 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 same conference by van Oorschot.

24. Colin Boyd and Wenbo Mao. Designing Secure Key Exchange Protocols. In Diete R. Gollmann, editor, Computer Security—ESORICS '94, pages 93–106. Springer-Verlag, 1994. This paper takes the view that program derivation techniques should 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 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 the 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, protocol using a trusted party and conference key protocols).
The paper deals with issues not addressed with (such as key revocation, the effect of mistrust towards trustworthy principals and more complex distribution protocols).

The formal mode is currently needed some adjustment. The problem lies with the definition of security for state of the system. While it would appear true that given a suitable initial state and the definition of the send and receive operations the resulting operation will be "secure"—then the notion of a suitable definition of initial state must be addressed. It is effectively feasible to setup an initial state (and generally such an initial state will not be a pathology alone, i.e. it will have some keys) that is useless. For example, a state where the only pair in the whole system is ($k_1$; 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 total to derive protocols rather than post-hoc verifying them.


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 Okamoto-Shiraishi signature schemes, RSA and others. It also addresses the Data Encryption Standard.


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 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 recognize 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 be legitimately derived from the successful execution of the protocol.

Some aspects of authentication protocols have been deliberately ignored (no attempt to cater for authentication of trustworthy principals and no analysis of encryption schemes strength).
The author is 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 criticized.

The author then gives informal accounts of some important notions in authentication:

- If you've seen Joe anumber that you've 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 or originate recently—indeed, after your turn.

- 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 author presents their formalism based on a many-sorted modal logic. Messages are guarded as statements in the logic. There are principals, keys, and formulas. A number of logical axioms are given.

The author assumes explicitly that a principal to detect and ignore a message he hasn't sent. The logic monotonic within a protocol run (that is, belief that holds at the start of a protocol run). Moreover, the logic assumes that a principal utter a formula \( X \) then he actually believe it.

The author states "each encrypted message contains sufficient redundancy to be recognized and decrypted unambiguously." This idea of recognizability will be taken up by other authors. Indeed, the notion itself is subtle and important. The notation of omitting these end of message is oftentimes used. It is, of course, the case, that decryption will be necessary if the actual ending of a message is to be known: that is, the message must have authenticity.

The author then presents a set of postulates fairly modestly: "we do not present the postulates in the most general possible form; our main concern 77
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 notation of trust involved. Then the nature of the trust is not made explicit. From the examples, however, it can be seen that trusting a principle to know a shared secret means that he will not reveal it himself; if there 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 $Q$. Multiparty 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 clear text. This idea will be taken up by other authors.

The author then presents the notion of an ideal 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 belief are omitted; thus there is no clear text in BAN messages. Each protocol is a sequence of encrypted formulae. The authors claim that their idealized formulae are clearer and more complete than other traditional descriptions. They also state that deriving an encoding from an idealized protocol is far less time-consuming and error-prone than understanding the meaning of a particular encoding. Omitting clear text gives rise to some problems, e.g. the direct leakage of information.

Loosely speaking, a message $M$ can be interpreted as a formula $X$ when every recipient gets them may deduce that these end must have believed $X$ when they 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 idealization. It seems that in addition to iteration of initial beliefs for the purpose of proof, suggested by the authors, one might well iterate over idealization too.

The protocol analysis takes the following steps:

1. The idealized protocol is derived from the original alone.
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.


E


3


4 11 BAN

O - R P
N - P
K - P
- F P
A RPC H
P
N - P K P
CCI .509 P ( )

12 13

M

( )

27 M B , M A , R M N , R N . AC
BAN 26

89 .

79
The authors state that Nesset's example "accurately points out an intended limitation or our logic" but indicates that the assumption by principal B that the published key is in fact good is contradictory. Though this is allowed by the formalism, it is not beyond the way man to notice. From this absurd assumption, Nessett derives an equally absurd conclusion.

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

---

### References


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


This contains draft proposals for various protocols.


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 are associated with probabilities. The paper finds bounds on various probabilities of interest using linear programming methods. Some examples are given.

We very much read.


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

- **Elementary Flaws**: Some protocols may provide only marginal protection, e.g., one that communicates passwords in clear, or the Nesset 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 freshness flaw in the Andrew Secure RPC protocol [26].

Orcle 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:

1. A → B: E(K_a: M)
2. B → A: E(K_b: E(K_a: M))
3. A → B: E(K_b: M)

After receiving the message at line 2, A decrypts with key k_a and as-sumes commutativity sends the result back to B in line 3. The single role or oracle flaw is for the intruder simply to pretend to be B and return M in line 2 and hence obtain M in line 3. The authors suggest that a "typing" check (to see whether the message is really an encrypted one) might solve this problem. As presented there is a more obvious problem that this will not work, namely there is nothing to stop the intruder simply acting as B using a key k_c that he knows in place of k_b. The protocol then works as normal.

Have I missed something? Typing flaws as a subclass of oracle flaws where in addition to using a fragment of the protocol as an oracle, the penetrator exploits his inability of all else to associate a word (or message) with a particular state of a particular protocol. Five different "types" of information can be distinguished:

- cryptographic protocols;
- protocol runs;
- transmission steps;
- message (sub)components;
- primitive types.

The paper then gives example typing flaws exposed in the Neumann-Stubbie protocol and the Otway-Rees protocol.
Cryptography related flaws arise due to the interactions of protocols and the particular cryptography methods employed.

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


The traditional method of specifying protocols has a well-defined syntax but no semantics. They tend very much to resemble their 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, nonce etc.), and to infer assumptions and goals. Internal actions are addressed too (for example, the presence of a word with type "nonce" implies 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 behavior 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 some means for overcoming some of the well-known deficiencies of the standard notation.


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 (there is a section to be commented on). 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).


In this paper, the author describes some attacks on recently published protocols highlighting assumptions about cipher block chaining 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).

82

This paper provides a summary of ways in which protocols fail and provide many examples of such flaws. Methodologies include: freshness attacks, type flaws, parallelsession attacks, binding attacks, and some implementation dependent attacks (e.g. Boyd's bit-flipping with stream cipher and cipher block chaining mishaps). Appraisals likely be left to other readers!

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 often wrong or otherwise incomplete. If predictable initial blocks are used then it will be possible for a principal to create the text for an arbitrary message of his choice.

John Clark and Jeremy Jacob. Freshness is Not Enough: Note on TrustedNonce Generation and Malevolent Principals. In this paper the authors demonstrate an unusual attack on a mutual authentication protocol described in Section 6.3.11. Malevolent choice of a nonce by one principal can cause a previously issued key to be accepted as fresh by the other principal.


In this paper the author argues that the DES algorithm is remarkably resilient to differential cryptanalytic attacks. This is because the method was known to the IBM design team in 1974. This should wake the reader up! What is in the public domain clearly lags well behind what is known to Government and the imageants. The criteria for designating the infamous S-boxes are described and discussed. An essential read for cryptanalysts everywhere.


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 still a useful book.

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 taken down: the initial key translation protocol makes assumptions about the content of the users 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 allows one of the servers to masquerade as the other.


1994 M A ( 6.9).

A

44 D M H N D C E E E
A

45

46 D E

1976.

46 . F P L P K M E E E
A C

1993.

47 P G , D J , J H , B R . S -
M CSP FDR: D B 2. T
, F S ( E ) L , A 1996.

C P/FDR

( )

). I

C P ( FDR2) L

I

( . . CBC, CFB )

48 P G , D J , B R . S M
CSP FDR: D B 3. T , F
S ( E ) L , J 1996.

47 .

CCI O

)

The goals of authentication provide the basis for much controversy. In this paper Golman 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 sometimes 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 use of encryption by protocol specifiers.


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. One advantage of this approach is 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 capability that could be offered.


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 which is then encrypted. A problem is that this provides a same means by which an attacker can verify keys she has guessed. Protocols that encrypt with weak keys, for example passwords, are vulnerable to guessing attacks. Implicit redundancy can only be recognized by the intended recipient(s) who knows the key for a particular example for a particular exchange. Examples are given.


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 86
only messages they can realistically expect. 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 belief implied by transmission of the message. Inference rules to accommodate the change are presented.


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

1. the supplier who creates the identifier;
2. the prover that inserts the identifier into a message; and
3. 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 time stamps, truly random numbers, counters, pseudorandom numbers, synchronized counters, and pseudorandom number generators and fresh encryption keys. The paper presents a table indicating which modes of use is secure for a particular approach indicating whether the prover is to be trusted or not. A brief categorization of messages replay is given.


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 a line of attack against the system. For such attacks to work here must be able to check whether such guesses are correct. The protocols 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 useful and worth reading.
Interesting extensions include the notion of recognizability (the use of typing would prevent many identified protocol flaws), the notion of possession (from a principal can possess, because he has seen them, and so on).

In distinction to BAN logic a principal does not have to believe in a for-mula in order to include it in a message (he merely has to possess it). Also included are explicit "not-originate here" indications for message components allowing the principal to detect plays of message she herself has created. There are also some message extensions by which precondition for actually sending a message are attached to it. The derivation rules given are much more numerous (over 40) than those given in BAN.


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-recognizer.

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. one explaining the disguising of super-increasing sequences in the knapsack problem). Merkle-Hellman and RSA schemes are outlined very briefly.


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 (V 4 and V 5) CHANGEPW and a critical examination (note that there is typographical error in the V 4 description). The question is raised as to what happened in the protocol when failure occurred.

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 88...
The authentication server while it stores the "old" password or the "new" password with the user depending on whether the previous attempts succeeded or not. Decrypting an appropriate ticket to give sufficient proof of identity. If the server already has carried out the update then there is no change and a successful 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 there of) to ensure that the ordering of the messages is deterministic (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 author(s). The proposed solution is quite though and certainly it seems sufficient. 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. 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] Tzollin 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 still there remains a flaw. [60] Tzollin Hwang, Narn-Yoh Lee, Chuang-Ming Li, Ming-Yung Ko, and Yung-Hsiang Chen. Two Attacks on Neuman-Stubble authentication protocols. Information Processing Letters, 53:103–107, 1995. This paper presents two attacks on the Neuman-Stubble protocol. The first is that given by Carlson [31] in 1994 (but note that his paper was submitted before Carlson’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 methods shown they suggest some alternatives (such as permuting the order of encryption to avoid replaying different messages). In fact it would appear that his approach is actually more secure since the protocol as it stands could be implemented using cipher block chaining. In that case replays become possible, with the replayed message just an 89.
i ni t i als e gm e nt o f th e f ir s tm e ss age; t hi s w ou l d n' t be t he c a se i f t h e r e w as s om e pl ai nt e xt pe r mu t at i on be f or e e nc r ypt i on.

The i m pr o m e n t f o r t h e s ub se quen t aut h e nt i c at i on pr ot oc o l al s o de p e n d s o n i m pl e m e nt at i on f o r s e c ur i t y . E x a m i n at i o n s h ow s t h a t i f ci p he r bl o c k c h ai n i ng i su s e d t h e r e i s a pr obl e m w i t h t h e i m pr o ve d s ol ut i on. Th u s , t h e of f e r e d s ol ut i on i s i m pl e m e n t at i on de pe nd e nt .

61 ISO/IEC. - E A -
1: ISO/IEC 9798

62 ISO/IEC. - E A -
2: E
ISO/IEC 9798
1993.

63 ISO/IEC. - E A -
3: E
ISO/IEC 9798
1995.

64 ISO/IEC. - E A -
4: E
ISO/IEC 9798
1993.

65 ISO/IEC. - E A -
5: E
ISO/IEC 9798
1993.

90

The paper provides a good introduction to beacon-based authentication. Conventions in cryptography are explained and an outline of Rabin’s (the original) approach to beacon use is given. A beacon emits 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 himself?) Simplifying, the parties exchange preliminary contracts and commitments to sign (conditional) followed by random integers $I_1$ and $I_2$. Let $I_e = (I_1 + I_2) \mod N$. They now exchange messages committing to the contract if the next beacon token is $I_e$. A party may not commit. If so, he has a $1/N$ chance of getting the other party at a disadvantage (and an $N-1/N$ chance of not getting away with it and having to explain his lack of commitment).

A beacon 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.


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.


A modern repeated authentication protocol. An initial protocol distributes as shared key $K_{AB}$ to principals $A$ and $B$ and also ticket $T_{AB}$; $A$, $K_{AB}$ generated by $B$ using a key known only to himself. 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 time stamps).
The author points out that there are 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 an alternative interpretation of "run off the protocol," namely that a run is initial protocol and all subsequent runs are off the repeated authentication protocol. It would appear that BAN already currently stands does not handle repeated authentications using tickets.

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

References:

- 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 Jospe specificational language (and so represents a use of a well-known general specificational notation for protocols specificational analysis and synthesis). The modern trend has been away from off-the-shelf technologies but this may change.


- V. Kessler and G. Wedell. AUTLOG — An advanced log 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 recent said prediction as originals 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 author also introduces the notion of a passive eavesdropper. This can be used to detect certain types of flaw (such as the Nessett flaw 92).
The paper also includes a discussion about the inability of the logic to handle parallel runs.


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 worryingly definite.


This paper provides a brief but wide-ranging bibliography of seventy-one papers on authentication. Its survey of the field in terms of goals of authentication, design aspects of cryptographic protocols, protocol categorization (private, public, hybrid, one-way functions, etc.), and verification of protocols. There is an adequate of where to find relevant information on various protocols. The column indicating which protocols are flawed is essential.


Seventeen years after publication of the Needham-Schroeder Public Key Protocol, Lowe discovers whatever 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.


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. He presents an account of how principal and intruder communications are modeled in CSP (with a restricted number of principals) and presents an argument to show that the analysis performed is insufficient to guarantee its correctness when more principals are added to the system.

The paper records a number of attacks upon protocols. The aim is largely to show that the same mistakes in protocol design are being made again and 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 a message she has created. Woo and Lam actually state that reflections are detected by principals and so the protocol has no means of enforcing this. Low believes that such functionality should be captured by the protocol and not left as an implementation dependency. Attacks on the KSL protocol 6.5.3 and on the TMN protocol are given.

---


In this paper the author examines some of the deficiencies of BAN logic. After noting that BAN logic passes as secure some patented and insecure protocols, they address some specific weaknesses. First, the ideal cases is examined (does not take into account context-specific information), then the nature of belief is examined (such as the senseless nature of believing in a nonce). The elicitation of assumptions is also a very difficult area. The author then goes on to provide their own formalism intended to cater for flaws identified. An element of pre-processing is carried out to identify the implicit use in the protocol script of various elements (e.g. nonces are identified as challenges, responses are identified etc.). A set of BAN-like inference rules are given. Two protocols are then analysed using the system.

---


94
This paper addresses two important points regarding authentication protocols. The first is that non-secret data is often encrypted by a principal or department 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 too much cryptographic 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 use of redundancy, which can lead to significant provision of cryptographic information.

The authors state that the use of a single notation for all cryptographic services gives a lack of precision that has led to many weaknesses and provides a notation that distinguishes between encryption for confidentiality and one way service.

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

The way CBC works allows CI and PI + CI (i-1) to become ciphertext pairs. Since these session keys differ in each run, this merely ensures that different runs allow different ciphertext pairs to be created.

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


A very good introduction to the history, terminology and theory of cryptography. 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 usual algorithmic fare.


Meadows provides a comprehensive account of the use of formal uniqueness in the development of security protocols. As well as given a survey so indicated 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 a very low level. Formal techniques are an important tool but information guidance from the user is needed to reach these stages automatically so supported is appropriate. An important tool.

83 Jonathan Mille.


The Introg plays an important part in the development of tools for protocol analysis. The user specifies the protocol in a prelog-based syntax and can use the tool interactively to determine whether specific states can be reached or specific data items compromised. It 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 needed to reach these stages automatically.

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


The use of particular algorithms in conjunction with particular protocols can have disastrous results. This is not because of any inherent weakness in the cryptographic algorithms themselves, rather it is because the way in which they are used require 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 are indicated used 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 failure are explained; a lottery 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 of specifiers do not state the precise qualities they demand of cryptographic algorithms.

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


This paper provides a brief overview of the development of modal logic and its 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 belief soft the form 

\[ B(p) \text{ unless } B(q) \]

is assumed to be true unless refuted by another evidence. An example application of the logic is given. Principal s wishing to communicate ask a server for a key to be distributed. A characterization of knowledge and belief about the protocols is recorded in 18 axioms. These axioms encompass rules about beliefs as a result of sending and receiving 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 "\( k \) is a key" for all \( k \)? (Axiom 1 starts to look strange in this context).

85 J H M P F C


86 L E M A L K B C

S I J T H C S

A few examples consequence of the axioms are given. The text includes that there is a need for quantification to be included in the logic.

Future plans include also the use of nested unless predictions. Combining the logic with a temporal logic is the final suggestion. Currently no tool supporting is available for the logic (but the author considers essential).

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


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 contain the classic crypt flaw [42]. It is usually referred to as "The Needham Schroeder Protocol" but it could equally apply to any of the three protocols presented in this paper. The authors were aware of many possible attacks and provided an effort 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 an approach of obtaining digital signatures via a third party using symmetric key encryption.

The authors assume that keys are not easily 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 technique 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.

In this paper the author revisits his famous conventional (symmetric) key authentication protocol and shows 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 to be provided to the authentication server. This is the non-included by the authentication server in the authentication ticket passed to B as part of the protocol, thereby ensuring freshness.


This brief paper (and its rejoinder) formed the start to 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 the they claimed (second order claim) and what the capabilities of BAN logic and its derivatives are (no claims, just investigatory science).

Nessett quotes (or misinterprets) the BAN authors’ statements regarding goals of authentication. This paper implies that the BAN authors had a particular position on what belief goals a protocol should have. 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 as a “well designed protocol that may have uses in certain environments”, even though the protocol does not achieve the goals Nessett states they regard as necessary (a point raised by Sivers on [107]).

Therefore an 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 encrypted with her principal 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 belief of both parties in the goodness of the key.

The paper is now part of the authentication folklore.


This paper gives an alternative protocol to that given by Keene 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 signed by the authentication server, rather than by one of the principals.
The paper criticizes the application of BAN logic to the KLS protocol, stating that it violates the notion of freshness. Further modifications to this protocol are suggested. The trade of involving particular approaches to authentication (e.g., timestamps or nonces) are examined.

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

D. O. T. M. A.

1987.

D. O. R.

(BAN 26).

R. R. L. A. M. O. D. S. P. K. C. C.


R. M. G. A.


A. G.

A.

A.

A. O.

(BAN).

C. C P. FDR

R. P.

DERA.

This report summaries how CSP can be used to model principal's 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 analyzing security protocols. The work was carried out under the Strategic Research Plan managed by the DERA.


This paper provides a review of existing literature in the field of authentication. The paper begins with some introductory definitions and a description of the Needham-Schroeder protocol, its flaws and their resolution by time stamps. 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.


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


This paper addresses issues how confidentiality and authentication of messages can be addressed within the CSP framework. The paper provides a brief introduction to CSP and can be read with little knowledge of the process algebra.

101
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 diagrams seem 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 177 MB/s when locked at 35 MHz.

Proprietary. 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.

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.

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.

Snekkens 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
<table>
<thead>
<tr>
<th>Page</th>
<th>Author(s)</th>
<th>Title</th>
<th>Venue</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>IEEE Computer Society Press</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P. B.</td>
<td></td>
<td>BAN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A. C.</td>
<td></td>
<td>HOL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HOL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IEEE Computer Society</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HOL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IEEE Computer Society</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HOL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IEEE Computer Society</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HOL</td>
<td></td>
</tr>
</tbody>
</table>
Section 2: Security Trust and Intentionality

The author categorizes the objectives of protocol analysis for both epistemic (knowledge) and doxastic (belief) logics. Loosely, belief is concerned with trust, functionality and legitimacy of a subject's viewpoint, whereas knowledge logics are used to investigate security and a penetrator's viewpoint.

Though intuitively epistemic logics seem more appropriate for security and doxastic logics for trust, each formally capable of capturing and reasoning about 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' critique of BAN. Syversen 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 goal of authentication are considered a source of confusion. Nett'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 K is good" is criticized, because such demonstration is known to be impossible in general. He goes on to give an example where second order belief are insufficient and concludes that the degree of belief demonstrated by a protocol varies according to the application.

The author also takes time to ask Cheng and Gligor for serious attributions. He also examines Nett's claims and points out that in place 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 of our formal method helped bring to light".

It is maintained that BAN logic has been much misinterpreted and that in practice it has revealed 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 semantics is to provide a means of evaluating logic. 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 (indeed, they should be trivial). A further view on formal semantics is that it is an alternative view (diversity).

The author introduces possible worlds semantics offering this as a mean of exposing the necessity flaw.

Overall: The paper is well written and the words are readable.

This paper describes the two-part Neuman Stubblebine protocol and shows how it can be attacked. The attacks assume certain implementation dependencies (for example, that 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 can be attacked. A final protocol that incorporates elements of the KSL protocol is then presented. The paper concludes with an analysis of what the goal of the KSL and NS repeated authentication protocols were and of the utility of BAN logic for addressing repeated authentication.

A good paper, with some nice attacks.


Asyet unraveled. Hence completeness.


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


An early protocol security classic. This paper describes attacks on communication protocols and measures that can be taken to counter them. Best of all is the low-level details of cryptosystem usage (in particular, the consequence of the use of particular approaches to the choice of initialization vector in DES). Essential reading.

Efficient DES Key Search, Crypto 93, August 1993.
This paper addresses considerable detail for a pipeline search machine for DES. A very good paper. Probably the most detailed hardware paper to ever reach a conference ever!


This is a simple introduction to cryptography assuming no math whatsoever. It gives a good introductory account to the history of cryptography and introduces various types of encryption algorithms. It is, of course, old. It takes the reader from Caesar cipher to DES and public key cryptography (but no detail on the latter).


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


This paper provides a good introduction to some principles of authentication, explaining some basic cryptography, what the system at stake is, what sorts of parties may wish to carry out authentication exchanges. The paper provides some paradigm 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 needed augmenting: the principal P needed to be included to precede the principal Q in steps 5 and 6. Some of the protocol shows are susceptible at least less. Indeed, Woo and Lam themselves published a correction to another protocol.


A previous paper by the authors described a protocol that was subsequently found to be flawed. The authors explain how they started with a secure (but laborate) 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 Principle of Full Information, which dictates that the initiator and responder include in every outgoing message all of the information that has been gathered so in the authentication exchange. A number of simplification heuristics are given. These are demonstrated...
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. An parallel session attack can be mounted to enable a malicious agent to start and complete an authentication exchange with—without the principal's identity being claimed once an authentication has taken place.


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


This paper provides a good introduction to some relevant issues for authentication protocols. It provides an overview of the historical 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 protocol 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 idea is that protocols 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, crypto/difference computing functions and making comparisons and simple decisions). Note that the approach taken is that a protocol does not require an notion of looping (effectively only assignment and alternation are needed).

A CPAL (C P A L ) BAN

CPAL

( / )

( )

( )

108
With the simulated execution model, the idea is that the goal of the protocol is stated as a post condition and the new p-value is used to derive the preconditions for success of the protocol (i.e., the initialization 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 Schako Private Key Protocol, and the Otway and Rees Private Key Protocol).

The ideas expressed in this paper appear useful but the particular section seems a little


This paper provides a rebuttal to Boyd and Mao’s paper at the same conference [23]. Oorschot maintains (with justification) that BAN passes the first protocols imply because the formal assumption of trust in the authentication server is not actually true. The second example protocol passed by BAN because the idea is simply wrong.

120 P C O A A E BAN L
. I T H , , E 93 , 765

B M
BAN 23. O
( ) BAN
.
BAN

109