

# Design and Analysis of Key Exchange Protocols via Secure Channel Identification\*

Colin Boyd and Wenbo Mao

Communications Research Group  
Electrical Engineering Laboratories  
University of Manchester  
Manchester M13 9PL, UK  
Email: Colin.Boyd@man.ac.uk

**Abstract.** We suggest a new methodology for design and analysis of key exchange protocols. The basic idea is to establish the minimum cryptographic requirements in delivering a new session key, and to identify how these are achieved in the protocol under examination. The method is therefore limited to key exchange protocols and to establishing that the basic properties exist. The method is easy to carry out by hand (although some existing protocols may be difficult to handle). It allows existing protocols to be re-designed and new protocols designed in a flexible manner. A number of new protocols designed with the method are suggested.

## 1 Introduction

There have been great advances in recent years in the understanding of how to design and analyse cryptographic protocols for authentication and key exchange. Such protocols are typically the initial step in setting up a secure communications session; they are therefore critical in ensuring that security is maintained. Furthermore, it has been found that design of these protocols is extremely error-prone and a methodical approach is important.

The two major classes of analysis technique are algebraic state based approaches and logical approaches. While there have been successes with both approaches each has its limitations and research continues into improving them. The algebraic approach [9, 14] is to model the system as a state-based machine, to define what are the bad states and show that they cannot be reached under the assumptions of the protocol. A difficulty of this approach is how to fully define the abilities of the attacking process and what exactly constitutes a bad protocol state. In addition there is a need for a computer tool to search the state space. The logical approach [1, 13] models the beliefs and/or knowledge of each protocol principal and uses global inference rules to try and achieve the desired final beliefs after the protocol is complete. Difficulties here include

---

\* This work is funded by the UK Engineering and Physical Sciences Research Council under research grant GR/G19787.

how to translate between the logical language and the usual manner of protocol description.

In addition to the issues mentioned above, all the existing methods are confined to protocol analysis and are of little help in re-designing a flawed protocol or designing a new one from scratch. While they may find bugs, it may or may not be obvious how the bug may be fixed, and one will certainly want to re-run the method with the altered protocol. It is therefore desirable to have methods that allow protocols to be designed in a methodical way that gives confidence that they are going to be correct. While it may be expected that such a method will restrict the domain from which such protocols may be chosen, this is likely to be out-weighed by the benefits. It is the aim of this paper to explain such a method. We will restrict ourselves here to illustrating the new method and giving an intuitive idea of the way that it works. A mathematical foundation is given in a companion paper [4].

In the remainder of this section we outline the method we are proposing. The following section examines a variety of protocols based on conventional symmetric cryptography as well as interactive key exchange. For each case we analyse a typical example and propose alternatives.

### 1.1 Secure Channels

Key exchange protocols make use of cryptography to ensure that the distributed keys are delivered to their destination in a manner that maintains their confidentiality and ensures their integrity. There are, however, a variety of different cryptographic functions available and it is important to select amongst them such that the correct security services are provided. Many cryptographic protocols use either conventional symmetric cryptographic algorithms or public key cryptosystems, while others make use of key agreement procedures, originally conceived by Diffie and Hellman [8]. Cryptographic security properties can broadly be classified into two types.

- **Confidentiality** allows the sender of a message to decide those users who will be able to receive it. These recipients will be defined by their possession of the required decryption key.
- **Authentication** allows the recipient of a message to decide those users who must have sent it. These senders will be defined by their possession of the required encryption key.

We can define abstract security channels, of confidentiality or authentication type, between pairs of users in possession of keys which allow cryptographic confidentiality or authentication to take place between them. These channels can be used to provide the security services needed to transport keys to be exchanged; furthermore, since these are defined in an abstract manner the particular cryptographic algorithms used can be left unspecified. We write

$$S \xrightarrow{c} A : m$$

to denote that  $S$  sends the message  $m$  on a confidentiality channel from  $A$  and

$$A \xleftarrow{a} S : m$$

to denote that  $A$  receives the message  $m$  on an authentication channel from  $S$ . Note that the common, informal, manner of expressing protocols uses the notation  $A \rightarrow B : m$  to express that the message  $m$  is transferred from  $A$  to  $B$ . However, it is *a priori* completely unknown to  $A$  who will receive  $m$  or to  $B$  who sent  $m$ . This is a quite different situation from the use of logical secure channels where confidentiality to the recipient and/or authentication from the sender is assured. In this paper we will continue to use the established notation to express concrete protocols where the security channels have been described.

In a normal key exchange protocol the aim is for two or more users to share a key for a subsequent session. It is possible for a trusted server to generate the key, or for one or more of the users to do so. Whichever of these is the case, the key needs to be passed to each user over an insecure physical channel. We will make an assumption, which is true in all practical cases, that the session key will be a shared key for use with a symmetric cryptosystem. However we will allow the methods used to exchange the session key to use either symmetric or asymmetric techniques. Whenever such an exchange takes place the following two conditions must hold.

1. The key must not be allowed to become known to any users apart from those participating in the protocol or trusted servers. Thus there must be a confidentiality channel from the generator of the key to the recipient(s).
2. Each recipient must be sure that the key comes from an authorised agent and is a new key for use with the stated users. Thus there must be an authentication channel to each recipient of the key from the originator.

These conditions are uncontroversial and form the basis of what we will mean by a secure key exchange protocol. In practice further goals are often desired, for example that each user has confirmation that the other users possess the new key. These additional requirements can always be achieved by additional steps independent of the actual key exchange.

To see the power of even these simple observations consider an initial example. This is the protocol of Tatebayashi, Matsuzaki and Newman (the TMN protocol) which is well known to be flawed and has been extensively analysed [9]. The protocol has the usual principals which are users  $A$  and  $B$ , who wish to share a new session key,  $K_{AB}$ , and a trusted server,  $S$ . Somewhat unusually,  $K_{AB}$  is generated by  $B$  and not  $S$ .  $S$  has a public key,  $K_S^{-1}$ , which is known to  $A$  and  $B$ . The messages exchanged in a successful protocol run may be defined as follows.

#### TMN Protocol

1.  $A \rightarrow S: A, B, \{R_A\}_{K_S^{-1}}$
2.  $S \rightarrow B: A$

3.  $B \rightarrow S: \{K_{AB}\}_{K_S^{-1}}$
4.  $S \rightarrow A: K_{AB} \oplus R_A$

Here, and throughout, the notation  $\{X\}_K$  indicates the string  $X$  encrypted using the key  $K$ . It can be seen that the initial channels in the system are defined by the public key. It is implicitly assumed that the purpose of this public key is for  $A$  and  $B$  to pass messages confidentially to  $S$ . Thus the two existing channels are

$$A \xrightarrow{c} S$$

and

$$B \xrightarrow{c} S$$

(If the public key were also to be used for authentication in the form of signatures, then it would define two further channels  $A \xleftarrow{a} S$  and  $B \xleftarrow{a} S$ . Existence of these extra channels does not alter the analysis.) By our conditions above,  $K_{AB}$  must be received by  $A$  on an authentication channel. But it can easily be seen that no such channel is used (or indeed exists), since  $A$  will accept any value sent in message 4. Thus there is no need to examine the protocol further since we can see that the correct properties have not been achieved<sup>2</sup>. It is not difficult to go from this observation to form a concrete attack, but even if we could not see such an attack we should still not be happy to use this protocol.

## 1.2 A Generic Secure Protocol

A variety of protocols may be defined by changing the way that the security channels are defined. In this paper we analyse previously published protocols by establishing how the channels are defined, even though they may not have been specifically identified by the authors of the protocols. We also define our own variations by forming these channels in the simplest way we can, with the aim of making the analysis as transparent as possible and at the same time removing all unnecessary processing. The above principles suggest a generic protocol for a key generator  $S$  to pass a key  $K$  to  $A$ , to be shared with a set of users  $\mathcal{U}$ , which can be abstractly defined as follows.

- $S \xrightarrow{c} A: K$
- $A \xleftarrow{a} S: K, N, \mathcal{U}$

The item  $N$  is a *nonce* which is intended to show that  $K$  is a new key and not a replayed one.  $N$  is usually either a random challenge previously sent by  $A$ , or a timestamp. A number of published protocols conform to this format and it is typical to combine both the authentication and confidentiality functions together in one cryptographic transformation. However this need not be so, and we will give examples where they are implemented separately. It should be noted that where this is done, it is implicitly expected that the key  $K$  should have its

<sup>2</sup> In this case it is possible to go further and show that the initial conditions are insufficient to succeed - no protocol can successfully transfer a shared key to  $A$  [2].

confidentiality protected on both channels. An obvious way to do this is via a public one way function.

It is intuitively easy to see that the generic protocol is secure in the sense that if the key generator  $S$  acts correctly then only users known to  $A$  (in other words those in  $\mathcal{U}$ ) may end up in possession of  $K$ .  $S$  is trusted to generate a new key for each protocol run and pass it on a confidentiality channel to exactly the set of users  $\mathcal{U}$ . Thus we may be sure that any protocol of this format will satisfy our basic security requirements as long as the cryptographic algorithms used are confidentiality and authentication functions as required. A more formal argument is given in a companion paper [4].

The generic secure protocol is the heart of the design and analysis technique we are suggesting. Protocol design proceeds by making the generic protocol concrete through choice of specific algorithms available in the target application. Analysis works by seeking to identify the secure channels used and comparing the message fields with those of the generic protocol. As our examples illustrate, where the channels used are difficult to identify, a re-design of the protocol frequently leads to simplifications.

## 2 Protocols using Symmetric Cryptography

In this section we examine what may be termed classic key exchange protocols as established in the seminal paper of Needham and Schroeder [11]. The principals in these protocols are the server  $S$  and two users  $A$  and  $B$  who wish to share a new session key which will be generated by  $S$ . The key will be distributed to  $A$  and  $B$  using keys  $K_{AS}$  and  $K_{BS}$  which are initially shared by  $A$  and  $S$ , and by  $B$  and  $S$ , respectively.

### 2.1 Needham and Schroeder Protocol

This protocol dates from 1978 and is well known to be flawed. The messages in a successful run are as follows.  $R_A$  and  $R_B$  are random challenges chosen by  $A$  and  $B$  respectively.

#### Needham-Schroeder Protocol

1.  $A \rightarrow S: A, B, R_A$
2.  $S \rightarrow A: \{R_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3.  $A \rightarrow B: \{K_{AB}, A\}_{K_{BS}}$
4.  $B \rightarrow A: \{R_B\}_{K_{AB}}$
5.  $A \rightarrow B: \{R_B - 1\}_{K_{AB}}$

The initial channels in the system are defined by the shared keys  $K_{AS}$  and  $K_{BS}$ . Depending on the algorithms used, these keys may define channels  $A \xrightarrow{c} S$ ,  $B \xrightarrow{c} S$ ,  $S \xrightarrow{c} A$ ,  $S \xrightarrow{c} B$  and  $S \xleftarrow{a} A$ ,  $S \xleftarrow{a} B$ ,  $B \xleftarrow{a} S$  and  $A \xleftarrow{a} S$ . When  $S$  sends the key he encrypts it using the shared keys defining the necessary confidentiality channels. When  $A$  receives the key in message 2, she has with it

a nonce and the name of  $B$ . This is similar to the generic form in section 1.2, the only difference being that the name of  $A$  is not included. In this case  $A$  can be sure that the message was intended for her as long as the algorithm used defines an authentication channel (that is,  $K_{AS}$  is required to form the message) and so we can regard  $A$ 's name to be implicitly included. On the other hand, when  $B$  receives the key he has no nonce with it. Thus he cannot be sure that the key is new. Since he needs to get this assurance from the key generator, the subsequent exchange will never convince him of it. This well-known problem can be extended to a concrete attack [6].

## 2.2 Otway and Rees Protocol

This protocol [12] was designed to overcome the known problems of the Needham and Schroeder protocol. The steps in a successful run of the protocol are as follows, using the same notation as above.

### Otway-Rees Protocol

1.  $A \rightarrow B: M, A, B, \{R_A, M, A, B\}_{K_{AS}}$
2.  $B \rightarrow S: M, A, B, \{R_A, M, A, B\}_{K_{AS}}, \{R_B, M, A, B\}_{K_{BS}}$
3.  $S \rightarrow B: M, \{R_A, K_{AB}\}_{K_{AS}}, \{R_B, K_{AB}\}_{K_{BS}}$
4.  $B \rightarrow A: M, \{R_A, K_{AB}\}_{K_{AS}}$

The value  $M$  is an extra random nonce chosen by  $A$  for practical, rather than security, purposes. It can be seen that there are many similarities with the Needham-Schroeder protocol. However, this time both  $A$  and  $B$  get to choose a nonce and have it returned with the key when it is delivered from  $S$ . The difference from the generic protocol suggested above is that the names of the recipients are not included with the key, in messages 3 and 4.  $A$  and  $B$  can therefore deduce that  $K_{AB}$  is new but cannot immediately deduce who else will know it.

Let us concentrate on message 4 received by  $A$ . As in the Needham-Schroeder protocol we may regard the name of  $A$  as implicitly included, because the authentication channel from  $S$  to  $A$  is symmetric. It is clearly intended that the returned nonce  $R_A$  is bound to the name of  $B$ , since they were sent together from  $A$  to  $S$  on an authentication channel in message 1. The exact logic by which  $A$  verifies that the key is to be used with  $B$  relies not just on the message received but also on the message sent to  $S$  and on the correct actions of  $S$ . It has been shown that the complexity of this argument is very sensitive to different interpretations of the exact actions performed [3].

We will not attempt to identify the exact authentication channel in this case, but instead note the protocol can easily be re-designed using the generic protocol. The resulting protocol is far easier to analyse since the purpose of each field is clear. At the same time the complexity of the protocol in terms of processing required is actually reduced.

### Modified Otway-Rees Protocol

1.  $A \rightarrow B: A, R_A$
2.  $B \rightarrow S: A, B, R_A, R_B$
3.  $S \rightarrow B: \{A, B, R_B, K_{AB}\}_{K_{BS}}, \{A, B, R_A, K_{AB}\}_{K_{AS}}$
4.  $B \rightarrow A: \{A, B, R_A, K_{AB}\}_{K_{BS}}$

Because this conforms to the generic protocol, its security is guaranteed as long as the encryption defines both confidentiality and authentication channels from  $S$  to  $A$  and  $B$  respectively. We note that, although we show the nonces being sent to  $S$  as plaintext, there are lower level arguments that suggest they should be encrypted [10]. This would not affect the analysis at this design level.

### 2.3 A New Protocol

Before leaving protocols based on symmetric cryptography we examine how to design a protocol which splits the confidentiality and authentication channels. This can, of course, be done in a variety of ways. Confidentiality may be obtained by any particular encryption algorithm such as DES. There is no need to assume any particular mode of operation and since the only data encrypted is a random key the simplest choice, electronic code book [5], may as well be used. Authentication may be achieved by a key dependent message authentication code (MAC) which could also be based on a block cipher or could use a suitable one-way hash function. It is important that the MAC also provides confidentiality of the key. Thus we are led to the following protocol.

#### A Split Channel Protocol

1.  $A \rightarrow B: A, R_A$
2.  $B \rightarrow S: A, B, R_A, R_B$
3.  $S \rightarrow B: \{K_{AB}\}_{K_{BS}}, MAC_{K_{BS}}\{A, B, R_B, K_{AB}\},$   
 $\{K_{AB}\}_{K_{AS}}, MAC_{K_{AS}}\{A, B, R_A, K_{AB}\}$
4.  $B \rightarrow A: \{K_{AB}\}_{K_{AS}}, MAC_{K_{AS}}\{A, B, R_A, K_{AB}\}$

There is a lot of flexibility in implementing this protocol. For example, it may be that different parts of  $K_{AS}$  and  $K_{BS}$  are used for confidentiality from those used for authentication. This way confidentiality may be implemented, say, with bits from a one-time pad to give unconditional secrecy to the keys. Another variation is to alter the physical paths of the messages, since in reality there are independent parallel conversations between  $S$  and each key recipient.

## 3 Diffie-Hellman Type Protocols

Protocols of Diffie-Hellman type [8] have a number of potential advantages over more conventional protocols. Firstly, there is no requirement for the presence of an on-line server to aid in the key distribution. (In compensation, users need

to have some mutual verification information order to provide key integrity, or provide it in terms of a public certificate.) Secondly, the users themselves participate in generation of the key. This makes it easier for them to be sure that the key chosen is new and not an old one being replayed. Finally, users do not need to share secrets with any other party, before the start of the protocol.

The generic protocol of section 1.2 does not at first appear to fit the situation of key agreement because there is not a single key generator involved, but instead the key is generated by the users themselves. However, there is no assumption that the key generator is not also a key user, and all that is required is to change the viewpoint of what messages mean in the Diffie-Hellman protocol. The Diffie-Hellman protocol uses a prime  $p$  and a primitive element  $\alpha \bmod p$  which are fixed for the system.

- $A$  chooses a random value  $x$  uniformly modulo  $p - 1$  and calculates  $R_A = \alpha^x \bmod p$ .  $A$  sends the value  $R_A$  to  $B$ .
- $B$  chooses a random value  $y$  uniformly modulo  $p - 1$  and calculates  $R_B = \alpha^y \bmod p$ .  $B$  calculates the session key  $K_{AB} = R_A^y \bmod p$ .  $B$  sends  $R_B$  to  $A$ .
- $A$  calculates the session key  $K_{AB} = R_B^x \bmod p$ .

In its basic form the protocol provides no authentication, to either user, of who possesses the shared key. This is a widely recognised limitation and a number of augmented protocols have been proposed to provide authentication. The station to station (STS) protocol was designed by Diffie, van Oorschot and Wiener [7] and they provided an informal analysis of the protocol security. Subsequently van Oorschot [13] analysed the protocol using an extension of an established logic which confirmed the conclusions previously reached by the original authors.

It is assumed that users can generate signatures and for this purpose certificates will need to be available either from a directory or in the protocol itself. However, we omit these in the following description. We denote the signature of user  $U$  on the message  $X$  by  $Sig_U(X)$ . In addition use is made of a symmetric encryption algorithm. A successful protocol run proceeds as follows, using the same notation as above.

### STS Protocol

1.  $A \rightarrow B: R_A$
2.  $B \rightarrow A: R_B, \{Sig_B(R_B, R_A)\}_{K_{AB}}$
3.  $A \rightarrow B: \{Sig_A(R_A, R_B)\}_{K_{AB}}$

When  $A$  receives message 2 she can calculate  $K_{AB}$  using  $R_B$ , and hence decrypt the second part. She checks that the result is the signature on  $(R_B, R_A)$  and if not then she aborts the protocol. Similarly  $B$  decrypts message 3 and checks that the result is  $(R_A, R_B)$ . If not he aborts the protocol, otherwise he proceeds to the session.

Let us now consider what secure channels are used to deliver and authenticate the key  $K_{AB}$ . We regard the value  $R_A$  as being  $K_{AB}$  encrypted with the value  $y$ . While this may seem an odd viewpoint, it is quite clear that the value  $y$  is a secret key that is required to recover  $K_{AB}$  from  $R_A$ , which is exactly the property of a confidentiality channel. The unusual aspect is that at the time of sending,  $A$  has no idea of the value of  $K_{AB}$  or who will receive it, except that the recipient will be any user who knows the value  $y$  such that  $R_B = \alpha^y$ . We call this user  $owner(y)$  and define the confidentiality channel as follows.

$$A \xrightarrow{c} owner(y) : K$$

means

$$A \rightarrow B : \alpha^x \text{ mod } p$$

where  $K = \alpha^{xy} \text{ mod } p$ . Note that in the concrete version we need to make some assumption about the recipient and so we write it down as  $B$  even though we cannot be sure who will get the message.

Now it is clear that for  $A$  to achieve authentication of the key it is only necessary for her to know that  $B$  knows  $K_{AB}$ , since the only user who may be in possession of  $K$  is  $owner(y)$ . This will therefore establish that  $owner(y) = B$  and so  $K$  has been sent to  $B$  on a confidentiality channel and received from  $B$  on an authentication channel. In the STS protocol we conclude that

$$B \xleftarrow{a} A : K$$

means

$$A \rightarrow B : E_K(\text{Sig}_B\{R_B, R_A\})$$

where the values  $R_A$  and  $R_B$  are defined as above. We can now see that it is an assumption in the STS protocol that it is necessary to know  $K$  in order to form  $E_K(z)$  from a given  $z$ . This is an implementation detail but it should be made clear that this property is absolutely necessary, since it is not held by all commonly used crypto-algorithms and unjustified assumptions of this sort can lead to protocol weaknesses [10]. While van Oorschot makes this assumption clear in his analysis [13] it only seems to be implicitly defined in the original definition of the STS protocol [7].

The rather complex nature of the authentication channel, which uses the signature algorithm, the key generation process and the unspecified encryption algorithm, contrasts with the simpler channel we have seen above. However, there seems to be no advantage in using this complex channel and therefore we suggest an alternative protocol with messages as follows, where  $h$  is a publicly known one-way hash function.

### Modified STS Protocol

1.  $A \rightarrow B: R_A$
2.  $B \rightarrow A: R_B, \text{Sig}_B(A, B, h(K_{AB}))$
3.  $A \rightarrow B: \text{Sig}_A(A, B, h(K_{AB}))$

While we do not believe the STS protocol is in any way unsound, we prefer the above protocol for two reasons.

- The authentication channel is much clearer. Each user receives a signed copy of the key and an acknowledgement of who the key is shared with. Notice that each user is already aware that  $K_{AB}$  is a new key since each participated in its generation using a new random value.
- The protocol is slightly simpler since the encryption algorithm is not required. Perhaps more important than efficiency is that fewer mechanisms are relied on. Note that it is usual to employ a hash function anyway when signing a message, and so the hash function  $h$  does not really require any additional processing.

## 4 Conclusion

We have identified the security channels used in previously published key exchange protocols. This has enabled us to suggest minor improvements and simplifications in these. We have shown how to design new protocols by using a generic protocol which can be made concrete in a number of ways. The techniques can easily be used by hand since they force a simplified representation of the protocols.

The techniques can be used in a variety of further situations. For example protocols based solely on one-way functions may be analysed and conference key protocols can also be accommodated [4]. Further research is needed to investigate extensions to multi-user and other more complex protocols.

The usefulness of the method can best be assessed by critical examination of the protocols which it is used to design. We therefore invite other researchers to assess the new designs suggested in this paper.

## Acknowledgements

This paper was written while the first author was a visitor at the Computer Science Department, University of Waikato, New Zealand.

## References

1. M.Burrows, M.Abadi, and R.Needham, *A Logic of Authentication*, Proceedings of the Royal Society, Vol A426, pp 233-271, 1989.

2. Colin Boyd, *Security Architectures using Formal Methods*, IEEE Journal on Selected Areas in Communications, June 1993, pp.694-701.
3. Colin Boyd and Wenbo Mao, *On a Limitation of BAN logic*, Advances in Cryptology - Eurocrypt 93, Springer-Verlag, 1994, pp.240-247.
4. Colin Boyd and Wenbo Mao, *Designing Secure Key Exchange Protocols*, Proceedings of ESORICS 94, Springer-Verlag, 1994, pp.93-105.
5. D.W.Davies and W.L.Price, *Security for Computer Networks*, John Wiley and Sons, 1989
6. D.E.Denning and G.M.Sacco, *Timestamps in Key Distribution Protocols*, Communications of the ACM, 24,8,1981, pp.533-536.
7. Whitfield Diffie, Paul C. van Oorshot and Michael J. Wiener, *Authentication and Authenticated Key Exchanges*, Designs, Codes and Cryptography, 2, pp.107-125, 1992.
8. W.Diffie and M.Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, IT-22, 6, November 1976, pp.644-654.
9. R.Kemmerer, C.Meadows and J.Millen, *Three Systems for Cryptographic Protocol Analysis*, Journal of Cryptology, 7,2, Spring 1994, pp.79-130.
10. Wenbo Mao and Colin Boyd, *Design of Authentication Protocols: Some Misconceptions and a New Approach*, Proceedings of IEEE Computer Security Foundations Workshop VII, 1994.
11. R.M.Needham and M.D.Schroeder, *Using Encryption for Authentication in Large Networks of Computers*, Communications of the ACM, 21,12, December 1978, pp.993-999.
12. Dave Otway and Owen Rees, *Efficient and Timely Mutual Authentication*, ACM Operating Systems Review, 21,1,pp.8-10, 1987.
13. Paul C. van Oorschot, *Extending Cryptographic Logics of Belief to Key Agreement Protocols*, Proceedings of the 1st ACM Conference on Communications and Computer Security, Fairfax Virginia, November 1993.
14. Darryl M. Stahl, Stafford E.Tavares and Henk Meijer, *Backward State Analysis of Cryptographic Protocols Using Coloured Petri Nets*, Workshop on Selected Areas in Cryptography, Canada, 1994.