DAY TWO
FRIDAY, MAY 6

USING FORMAL METHODS TO ANALYZE ENCRYPTION PROTOCOLS

Richard A. Kemmerer

Reliable Software Group
Department of Computer Science
University of California
Santa Barbara, CA 93106

Extended Abstract

In this talk an approach to analyzing encryption protocols using machine aided formal verification techniques is presented. When considering a secure network that uses encryption to achieve its security (as opposed to using only physical security) one must consider both encryption algorithms and encryption protocols. An *encryption algorithm*, such as DES or RSA, is used to convert clear text into cipher text or cipher text into clear text. That is, the unencrypted message (clear text) is enciphered using a particular encryption algorithm to produce the unreadable cipher text. Similarly, the same or a symmetric algorithm is used to recover the original clear text message from the encrypted cipher text. An *encryption protocol* is a set of rules or procedures for using the encryption algorithm to send and receive messages in a secure manner over a network. The approach presented in this talk does not attempt to prove anything about the strength of the encryption algorithms being used. On the contrary, it may assume that the obvious desirable properties, such as that no key will coincidentally decrypt text encrypted using a different key, hold for the encryption scheme being employed.

The idea of the approach is to formally specify the components of the cryptographic facility and the associated cryptographic operations. The components are represented as state constants and variables, and the operations are represented as state transitions. The desirable properties that the protocol is to preserve are expressed as state invariants and the theorems that must be proved to guarantee that the system satisfies the invariants are automatically generated by the verification system.

The results of applying this approach to two different examples are discussed. The first example is a single-domain communication system using dynamically generated primary keys and two secret master keys, as described in [MM 80]. The host manages the communication keys for the system, and the terminals communicate directly with the host system. Whenever a terminal wants to start a new session with the host, the host generates a new session key. If this key were sent to the terminal in the clear a penetrator tapping the line could intercept the key and decipher all of the messages for that session. To prevent this each terminal has a permanent terminal key that is used by the host to distribute the new session key to a terminal when a new session is initiated. That is, a terminal's session key is the primary communication key for that terminal. It is dynamically generated by the host for each session. The static terminal key is the terminal's secondary communication key. It is used by the host to encrypt new session keys for transmission to the terminal.

Both the terminal keys and the current session keys are stored at the host. However, because a penetrator can be an authorized user, it is unsafe to store the terminal and session keys in the clear at the host. Thus, they are stored in encrypted form. There are two data structures of interest in the host: the terminal key table and the session key table. The *terminal key table* is static; each entry in this table contains the unique terminal key for the corresponding terminal encrypted using a secret master key KMH1. Unlike the terminal key table, the *session key table* is a dynamic structure. This table is updated each time a new terminal session is started; there is one current session key per terminal. Each entry in the table contains the current session key for the corresponding terminal encrypted using a second secret master key KMH0.

No terminal key, session key, nor either master key is in the clear in the host. To store the two masters keys a *cryptographic facility* is connected to the host. This facility may be accessed only through the limited cryptographic operations that are provided. The operations provided by the cryptographic facility are encipher data (ECPH), decipher data (DCPH), and reencipher from master key (RFMK).

The second example discussed is a protocol due to Tatebayashi, Matsuzaki, and Newman [TMN 91]. The Tatebayashi-Matsuzaki-Newman (TMN) protocol is a key distribution scheme by which a pair of ground stations in a mobile communication system obtain a common session key, through the mediation of a trusted server. Messages to the server are encrypted in the server's public key, which is known to all ground stations, and ground stations generate conventional keys to be used as session keys.

The TMN protocol works as follows. The server possesses a public-private key pair. The public key is known to everyone in the system, while the private key belongs to the server alone.

1. When user A wishes to communicate with user B, it encrypts a random number with the server's public key, and sends the encrypted random number, along with its name and the name of B.
2. When the server receives the request, it decrypts the random number and stores it as a key-encryption key for that conversation; it also notifies B that A wishes to start a session.
3. User B, on receiving the notification from the server, generates a random number to be used as a session key, encrypts it with the server's public key, and sends it to the server.
4. The server decrypts the response, encrypts the key with A's random number using a private-key algorithm, and sends the result to A. In the specification of the protocol, the algorithm used for this step is commutative in the sense that, if A[B] represents the encryption of B under the key A, then A[B] = B[A]. This could be done with a bitwise exclusive-or of the two keys, for example. The commutativity is significant for analysis.
5. User A decrypts the message from the server using the original random number it had generated and assumes that the result is the session key.

A more detailed discussion of the first example can be found in [Kem 89]. The details of the second example along with a discussion of three different approaches to analyzing encryption protocols can be found in [KMM 94].

An advantage of the approach presented in this talk is that the properties of a cryptographic facility can be tested before the facility is built. First, the system is represented using a formal notation. The resulting formal specification is used to generate the necessary proof obligations that assure that the system satisfies certain desired properties. If the generated theorems cannot be proved, then the failed proofs often point to weaknesses in the system or to an incompleteness in the specification. That is, they often indicate the additional assumptions required about the encryption algorithm, weaknesses in the protocols, or missing constraints in the specification.

Another advantage of this approach is that the cryptographic facility can be analyzed assuming different encryption algorithms by replacing the set of axioms that express the properties assumed about the encryption algorithms with a new set of axioms that express the properties of a different encryption algorithm.

References

[Kem 89] Kemmerer, Richard A., "Analyzing Encryption Protocols Using Formal Verification Techniques," *IEEE Journal on Selected Areas in Communication*, Vol. 7, No. 4, May 1989.

[KMM 94] Kemmerer, Richard A., C. Meadows, and J. Millen, "Systems for Cryptographic Protocol Analysis," *Journal of Cryptography*, Vol. 7, No. 2, 1994.

[MM 80] Meyer, Carl H., and Stephen M. Matyas, *Cryptography*, John Wiley, 1980.

[TMN 91] Tatebayashi, M., N. Matsuzaki, D. B. Newman, "Key Distribution Protocol for Digital Mobile Communication Systems," in *Advances in Cryptology - CRYPTO '89, LNCS 435*, G. Brassard, ed. Springer-Verlag, 1991.

ANALYSIS OF CRYPTOGRAPHIC PROTOCOLS I

A Taxonomy of Replay Attacks

Paul Syverson
Code 5543
Naval Research Laboratory
Washington, DC 20375
(syverson@itd.nrl.navy.mil)

Abstract

This paper presents a taxonomy of replay attacks on cryptographic protocols in terms of message origin and destination. The taxonomy is independent of any method used to analyze or prevent such attacks. It is also complete in the sense that any replay attack is composed entirely of elements classified by the taxonomy. The classification of attacks is illustrated using both new and previously known attacks on protocols. The taxonomy is also used to discuss the appropriateness of particular countermeasures and protocol analysis methods to particular kinds of replays.

The paper will be presented at the Computer Security Foundations Workshop VII, June 14–16, 1994, Franconia, New Hampshire. It will appear in the proceedings of that conference, published by IEEE Computer Society Press. Copies are available from the author.

Classification of Cryptographic Techniques in Authentication Protocols*

Wenbo Mao Colin Boyd

Communications Research Laboratory
Department of Electrical Engineering
University of Manchester
Manchester M13 9PL
UK

+ 44 61 275 4506 + 44 61 275 4562

wenbo@uk.ac.man.ee.comms colin@uk.ac.man.ee.comms

Abstract

In many published authentication protocols, the cryptographic services are coarsely specified in that, whenever needed, a uniform notation is used to denote them while the exact nature of protection required is left unclear. In this paper we reason that such a coarse treatment not only forms a foundation why authentication protocols are error prone, but is also responsible for a typical feature of misusing redundancy which causes many protocols to be unnecessarily weak. We propose new notations to refine protocol specifications. The refinement leads to a methodology for the development of secure and strong authentication protocols.

1 Introduction

For an authentication protocol which employs secret-key cryptographic techniques, the customary presentation is to write a few sequentially ordered lines. A typical line has the following form:

$$A \rightarrow B : M$$

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

This line describes a message directed from principal A to principal B. M denotes the transmitted message. In its simplest form M can be a single piece of text, e.g. the principal name A. More often, M is a structure of several sub-message components, e.g. M is A, N where the comma denotes the concatenation of the two sub-messages. A widely used notation for denoting a message part as a ciphertext is $\{M\}_K$ where M is the input data to a symmetric crypto-algorithm which is parameterised by the secret key K. There are other familiar notations for indicating the use of cryptographic protections, e.g. $eK(M)$ in an international standards recommendation [4], though the idea is the same: a protocol uses a uniform notation to denote the cryptographic protection wherever the data needs such protection.

1.1 Problem

What are the properties required of the cryptographic techniques in an authentication protocol? When speaking of the properties of a crypto-algorithm we refer to a collection of security services which include at least two distinct notions of cryptographic protection: *confidentiality* and *integrity*. A protocol designer should be aware that different kinds of protection are unevenly required by different parts of an authentication protocol. Perhaps it is assumed that the specified use of cryptographic services are *ideal* in that they supply any and all needed security services. This is a dangerous assumption for at least three reasons. Firstly, it is difficult for the protocol designer to precisely express what nature of protection is really needed and the vague expression of requirements makes protocol analysis far harder, which is equivalent to say that it is more unlikely to spot a subtle design flaw. In Section 2 we look at how such a flaw is introduced into a protocol design which has escaped a formal analysis. Secondly, it is difficult for implementors to understand how to realise the vaguely specified requirements and very possibly this will lead to a flawed implementation. In Section 3 we will see a wrong implementation suggested by two international standards organisations in a series of protocol standard recommendations [5, 6]. Thirdly, with the uniform notation it is difficult to make a sensible expression on the use of redundancy. Unthinking use of redundancy has led to many protocols, e.g. Kerberos [8, 7], to be unnecessarily weak in addition to being unnecessarily inefficient in execution. We reveal this problem in Section 4.

Having made the problems clear, in Section 5 remedies will be considered. These consist of new notations to refine the currently applied method of protocol specification and a methodology based on the refinement technique for the development of secure and strong authentication protocols. Our technique will then be used to guide the redesign of two well-known authentication protocols.

2 Vague Expression of Requirements

In this paper we will build our argument on a close study of an authentication protocol, the protocol of Otway and Rees [10].

$$
\begin{aligned}
&1\ A \rightarrow B : M, A, B, \{N_A, M, A, B\}_{K_{AS}} \\
&2\ B \rightarrow S : M, A, B, \{N_A, M, A, B\}_{K_{AS}},\ \{N_B, M, A, B\}_{K_{BS}} \\
&3\ S \rightarrow B : \quad M, \quad \{N_A, K_{AB}\}_{K_{AS}},\ \{N_B, K_{AB}\}_{K_{BS}} \\
&4\ B \rightarrow A : \quad M, \quad \{N_A, K_{AB}\}_{K_{AS}}
\end{aligned} \tag{1}
$$

This protocol uses a trusted authentication server S. Keys K_{AS} and K_{BS} are already shared between specific principals through some distribution method in a higher level of the security hierarchy. N_A and N_B are random numbers chosen by principals A and B, respectively, and have never been used before; such a number is called a *nonce*, meaning number to use only once [9]. Messages containing these two nonces in the first two lines form challenges sent to the server S, whose responses also enclose the nonces and thus the timeliness of the replied messages can be checked by A and B, implying that the server S exists recently. Hence the new key K_{AB} is generated recently by the server; this further tells A and B that each of their intended authentication objects, i.e. B and A, respectively, exists recently since otherwise the S would not have responded to either of them.

The protocol uses a uniform notation $\{\cdot\}_K$ to express requiring cryptographic services. The method is widely applied in the area of study. The uniform notation is understood to represent the use of some "ideal" cryptographic technique which can provide any and all cryptographic services needed. Under such a convention the protocol is secure as informally analysed in the previous paragraph.

However, we think that such a way of specifying an "ideal" cryptographic service forms an *incomplete specification.* In other words, the widely applied method for protocol specification is coarse; it does not support a sensible expression of the different kinds of cryptographic protection needed in different contexts of a protocol. For instance, in the first two lines of Otway-Rees protocol, the required nature of protection is message integrity, not that of confidentiality (as will be apparent later in this paper). The original designers did not make this requirement clear due to the inaption of the specification method. Below we look at how this coarse expression has diverted attention of a formal analysis from identifying the nature of the protection and resulted in a flawed recommendation for revision of the protocol.

The revision was suggested after the original protocol was analysed by the well-known formal logical analysis technique BAN [2]. The suggestion was to leave out the nonce N_B in line 2 from the specified protection. The idea here is that the nonce N_B in the protocol need not be secret; though sent by B in plain-

text, it is thought to still form a usual usage of challenge-response mechanism. Thus, in the revision we see line 2 as below:

$$2'\ B \rightarrow S:\ M, A, B,\ \{N_A, M, A, B\}_{K_{AS}},\ N_B,\ \{M, A, B\}_{K_{BS}}$$

Unfortunately, misled by the coarse specification in the original protocol, this revision resulted in throwing away the needed integrity service together with the unwanted confidentiality. The required integrity service in line 2 should be specified between N_B and the principal name A. Without this, the key K_{AB} can turn out to be shared between B and another person. An explicit attack on this revision is in [1]. In the following section we will devise a similar attack to a slightly more difficult situation: confidentiality rather than integrity of the message part $\{N_B, M, A, B\}_{K_{BS}}$ in the original protocol is protected.

3 Wrong Choice of Cryptographic Service

In the past few years, the International Standards Organisation (ISO) and the International Electrotechnical Commission (IEC) have been jointly developing standardisation of a number of entity-authentication and key-management protocols. Two parts of the standards recommendations (ISO/IEC 9798-2 and 11770-2 [5, 6]) propose a collection of protocols which employ symmetric cipher techniques. ISO/IEC specified these protocols with the conventional, widely applied method: a uniform notation is used to express the need of any and all cryptographic services in all protocols throughout the two documents. Besides the coarse specification, ISO/IEC further recommended, in a note, using *one* algorithm to implement the uniform notation (quoting from [5]):

"...ECB mode should not be used, while a chaining mode such as CBC with appropriate initialization values could be used".

Here, "ECB" refers to *electronic code book* mode of operation, the simplest block encryption method in which message blocks are encrypted separately. Such a mode is well known to be unsuitable for normal session encipherment. It is also well known that *cipher block chaining* (CBC) standard mode of operation is a suitable alternative removing the problem of ECB. CBC algorithm has been internationally standardised [3] and is one of the most common ways of implementing a block cipher. The mode has not only been chosen by ISO/IEC, but also by Project Athena in the current implementation of Kerberos [7].

At the end of this paper it will be clear that in general CBC should not be considered as a suitable candidate for implementing authentication protocols, though it will not be too harmful if the algorithm is used to protect some message part in a protocol where the required nature of protection is confidentiality. However, it will be *very harmful* if, as apparently intended by ISO/IEC, the algorithm is used

to implement the uniform notation throughout the protocol. We now explain a cut-and-paste property of CBC algorithm, which was first identified in [11].

The output of a block cipher using CBC mode is a sequence of n-bit cipher blocks which are chained together in that each cipher block is dependent not only on the current input plaintext block, but also on the previous output cipher block. Let $P_1, P_2, \cdots, P_m$ be plaintext blocks to be input to CBC algorithm and $C_1, C_2, \cdots, C_m$ be ciphertext blocks output from the algorithm. Then the encryption procedure to generate a block of ciphertext can be put as below:

$$C_i = eK(P_i \oplus C_{i-1})$$

where $eK()$ denotes an encryption algorithm keyed by K and $\oplus$ denotes the bitwise XOR operation. The first ciphertext block C_1 will be generated by using a random number block as C_0; this block is called initialising vector and denoted by IV. The decryption procedure to reveal a block of plaintext is as follows:

$$Q_i = dK(C_i) \oplus C_{i-1} \tag{2}$$

Here $dK()$ denotes the decryption algorithm which cancels the effect of $eK()$, i.e.

$$dK(C_i) = dK(eK(P_i \oplus C_{i-1})) = P_i \oplus C_{i-1} \tag{3}$$

Now substituting $dK(C_i)$ in Equation (2) with $P_i \oplus C_{i-1}$ we obtain:

$$Q_i = P_i \oplus C_{i-1} \oplus C_{i-1} = P_i$$

Consider now two messages divided into plaintext blocks as

$$plain\_blocks_1 = P_1, P_2, \cdots, P_l$$

and

$$plain\_blocks_2 = P'_1, P'_2, \cdots, P'_m$$

which are encrypted with the same key K using CBC algorithm as

$$cipher\_blocks_1 = IV_1, C_1, C_2, \cdots, C_l$$

and

$$cipher\_blocks_2 = IV_2, C'_1, C'_2, \cdots, C'_m$$

Suppose now that these two encrypted messages are cut into two and the first part of $cipher\_blocks_1$ is pasted onto the second part of $cipher\_blocks_2$. Thus the encrypted message becomes

$$cipher\_blocks_3 = IV_1, C_1, C_2, \cdots, C_i, C'_j, C'_{j+1}, \cdots, C'_m$$

where $1 \leq i \leq l$ and $1 \leq j \leq m$. If the legitimate recipient now attempts to decrypt $cipher\_blocks_3$ using key K, the message obtained is the following.

$$plain\_blocks_3 = P_1, P_2, \cdots, P_i, X, P'_{j+1}, \cdots, P'_m \tag{4}$$

Here block X does not correspond to any block from either $plain\_blocks_1$ or $plain\_blocks_2$ but instead we have (see Equations (2) and (3) above):

$$X = dK(C'_j) \oplus C_i = P'_j \oplus C'_{j-1} \oplus C_i \tag{5}$$

This particular message block will usually look random to the recipient. But certain plaintext blocks in authentication protocols, such as those containing secret keys may be *expected* to look random. This gives a strong hint as to how this property may be useful to an attacker. Furthermore, if the second message $cipher\_blocks_2$ is old, or if a plaintext/ciphertext pair can be found (in Section 4 we will see that protocols developed using the currently applied method are likely to generate plaintext/ciphertext pairs) then the value of P'_j may already be known to an attacker. Since C'_{j-1} and C_i are sent on the channel, the value of X may be found by the attacker. Applying this cut-and-paste technique, we have found that several protocols in the ISO/IEC standard recommendations 9798-2 and 11770-2 [5, 6] can be compromised. Below we use Otway-Rees protocol (similar to some protocols in ISO/IEC 9798-2 recommendation) as an example to demonstrate such an attack.

3.1 Attacking CBC-implemented Otway-Rees Protocol

First, data to be encrypted should be divided into a number of blocks. Let us focus on the message fragment $\{N_B, M, A, B\}_{K_{BS}}$ in message line 2. The following division may be expected:

$$P_1 = N_B,\ P_2 = M,\ P_3 = \cdots$$

where each P_i is an n-bit number block. Certainly alternative ways of dividing the message are possible but to show the attack is feasible we need only show that such a division is reasonable. We assume that one block has 64 bits, as in the case of DES and most other well-known block ciphers.

Making "N_B" into a single block is reasonable since a nonce should be chosen from a sufficiently large domain to guarantee uniqueness. Next, we expect "M" to occupy the whole block P_2 because allowing "M" to share the second block with the information item "A" is not convenient since "A" here stands for the address of a principal, which can have a variable length. Finally, we expect to start a new block for the rest of the message "$A\cdots$".

Viewed in the communication channel, cipher blocks of this message fragment are as follows:

$$cipher\_blocks_1 = IV,\ C_1,\ C_2,\ C_3, \cdots$$

where IV is the initialising vector. For this implementation an attack is possible. An enemy E first records the following pieces of data:

$$M', E, B,\ \{\cdots\}_{K_{ES}},\ \{N'_B, M', E, B\}_{K_{BS}}$$

E can record these blocks in the history when the protocol was running normally between himself and B. Let the message fragment $\{N'_B, M', E, B\}_{K_{BS}}$ correspond to the following cipher blocks:

$$cipher\_blocks_2 = IV',\ C'_1,\ C'_2,\ C'_3,\ \cdots$$

With these recorded blocks E can apply the cut-and-paste technique. E can ask B to send $cipher\_blocks_1$ above onto the channel by masquerading as A as follows:

1 $E \rightarrow B:\ M, A, B,\ \{\cdots\}_{K_{ES}}$

E intercepts the whole message sent from B to S and replaces blocks $C_2,\ C_3,\ \cdots$ in $cipher\_blocks_1$ with his recorded counterparts $C'_2,\ C'_3,\ \cdots$. Let $cipher\_blocks_3$ denote this altered cipher message:

$$cipher\_blocks_3 = IV,\ C_1,\ C'_2,\ C'_3,\ \cdots$$

Now E masquerades as B and sends the forged message line 2 to S as follows:

2 $E \rightarrow S:\ M'', E, B,\ \{N_E, M'', E, B\}_{K_{ES}},\ cipher\_blocks_3$

where the value M'' is carefully chosen by E as explained below. On receipt of this line, S will use the standard formula to decrypt $cipher\_blocks_3$ and retrieve

$$plain\_blocks_3 = N_B,\ X,\ E,\ B$$

where (applying Equation (5) above) $X = M' \oplus C'_1 \oplus C_1$. The value of X is known to E because he has each component on the right-hand side. Thus E should choose $M'' = X$ in the forged message line 2. In the rest of the protocol run, E should change M'' in message line 3 back to M and also intercept the message line 4 so that B will not see any difference from a normal protocol run. The result is that B and E will share a key that B thinks is shared with A.

From the above attack we realise that CBC algorithm does not provide the necessary protection. The cryptographic property that is needed in message lines 1 and 2 of Otway-Rees protocol is *integrity*, not confidentiality. Thus, an additional measure which provides such protection is indispensable. Practical ways include computing oneway hash functions and various checksum mechanisms. These computations are oneway; both sender and recipient compute them in the direction of encipherment using plaintexts as input.

Considering that the nonces and principal names in Otway-Rees protocol are not secret at all, the additional measure for the necessary integrity protection means that the CBC protection of these non-secret messages is purely superfluous. In next section we will further see that the superfluous protection of confidentiality on non-secret data is not desirable in a basic sense of security.

4 Unthinking Use of Redundancy

Undoubtedly the cut-and-paste attack described in the previous section can be avoided by taking special precautions. For example it will not work if the encrypted data (in particular the plaintext blocks $P'_{j+1}, P'_{j+2}, \cdots, P'_m$ in Formula (4)) contain sufficient redundancy, such as texts saying:

"this is the Otway-Rees protocol line number 2, run number 1005 in response to principal Alice's request of authentication toward principal Bob ..."

Such redundancy in the encrypted part of the data seemingly allow the recipient to check the data origin after decryption. Thus the computational methods for integrity check that we have suggested at the end of Section 3 may be saved. Indeed, ISO/IEC did recommend such a treatment. Every protocol proposed in 9798-2 and 11770-2 has a text field specified inside the encrypted message part. According to the specification these text fields are intended for decryption by the recipient. In Annex A of 9798-2 ISO/IEC gave clear guidance on how to use these text fields [5]:

"the encrypted text fields may be used to provide additional redundancy. Any information requiring confidentiality or data origin authentication should be placed in the enciphered part ... text fields may be used to state which entity requests the authentication".

We regard such a way of adding redundancy as potentially dangerous. In practice, cryptographic keys are classified as either key-ciphering keys or data-ciphering keys (session keys). A key-ciphering key is more expensive to establish than a data ciphering key, but once established, the former is long-lived while the latter is used for one session only. To add redundancy, especially the kind with a specific format, into the encrypted part of a protocol for later decryption means to use a key-ciphering key almost like a session key. The misuse forms a damaging element to these long-lived keys. In so doing each protocol execution will disclose statistical information for analysing the key-ciphering keys. Moreover, because the contents of such formatted redundancy is mostly public or easily guessable, in terms of information leakage we may expect a situation worse than that in a normal session of secret communication. For instance, let X be a block of known or guessable plaintext (e.g. part of a principal name) and let C_i, C_{i+1} be two consecutive cipher blocks generated by a block cipher in CBC mode where C_{i+1} corresponds to the plaintext block X; then noticing Equation (3) in Section 3, we have that $dK(C_{i+1}) = X \oplus C_i$ and C_{i+1} form a plaintext/ciphertext pair. Considering the long lifetime of a key-ciphering key and the high frequency of protocol executions, the known or guessable redundancy added for decryption may generate really *large* amount of plaintext/ciphertext pairs between given principals.

Our preference is to add *no* redundancy into the encrypted part of a pro-

tocol message where the message is intended for the purpose of data retrieval by decryption. Furthermore, we should not protect the confidentiality of any non-secret messages, or messages which contain formatted, predictable and statistical information. Examples of such messages include: names of principals and protocols, sequence numbers of executions and timestamps. Redundancy and non-secret data, when used for providing data origin check, should be put in where the message is to be transformed using a non-reversible, or oneway transformation. Oneway hash functions and some checksum mechanisms provide such a service. By taking these measures, the chance of generating information for use in cryptanalysis can be minimised.

5 A Refinement Approach

We have reasoned that the widely applied methods for protocol specification do not form a good mechanism in terms of helping either the designer to precisely express the requirements or the implementor to correctly understand what it is intended to achieve. In order to precisely express and correctly understand authentication protocols we propose new notations to distinguish between cryptographic transformations which provide confidentiality, and those which provide integrity. Let $\mathcal{X}$ and $\mathcal{Y}$ be message fields. Below we define two distinguished notions of message transformation between $\mathcal{X}$ and $\mathcal{Y}$.

$[\![\cdot]\!]_K$ *confidentiality*: with knowledge of K, the notation defines an algorithmic way to transform each $x \in \mathcal{X}$ to $[\![x]\!]_K \in \mathcal{Y}$ and to decide, for each $y \in \mathcal{Y}$, an $x \in \mathcal{X}$ such that $[\![x]\!]_K = y$; without knowledge of K, the notation does not define a computationally feasible way to decide $x \in \mathcal{X}$ from any given $y \in \mathcal{Y}$.

$[\cdot]_K$ *integrity*: with knowledge of K, the notation defines an algorithmic way to transform each $x \in \mathcal{X}$ to $[x]_K \in \mathcal{Y}$, but it does not define a computationally feasible way to deside $x \in \mathcal{X}$ from any given $y \in \mathcal{Y}$; without knowledge of K, the notation does not define a computationally feasible relation between $\mathcal{X}$ and $\mathcal{Y}$.

Below we list three principles for the development authentication protocols.

1) The designer should be aware what type of cryptographic protection is needed in which message fragment of the protocol and should use the new notations to specify the protection.

2) For transformations denoted by $[\![\cdot]\!]_K$, the sender generates it by encryption; the recipient retrieves data from it by decryption. Data input to it must be secret and should be designed to avoid containing, or to contain as little as

possible, any redundancy or predictable element. Specifically, timestamps, sequence numbers, and principal and protocol names should not be input to $[\![\cdot]\!]_K$.

3) For transformations denoted by $[\cdot]_K$, both the sender and the recipient generate it by encryption. The implementors should keep in mind that the chosen technique does not have any algorithmic method to retrieve data from $[\cdot]_K$ even with the key K; this property is to ensure not to form plaintext/ciphertext pairs.

These principles form a methodology for protocol development. Adhering to the rules, we are only allowed to express distinct cryptographic services, are required to justify use of cryptographic processing modes and must justify whether a certain type of data is entitled for certain types of cryptographic service. Misuses of services and keys become unlikely, or at least difficult.

It is our belief that an authentication protocol which has been developed in a non-methodical way can always be redesigned by following our methodology such that the protocol's functionality intended by the original designer preserves or gets improved. Here we demonstrate two redesign examples: the Kerberos protocol and Otway-Rees protocol.

The Kerberos protocol as presented in [2] is as follows.

1. $A \rightarrow S:\ A, B$
2. $S \rightarrow A:\ \{T_S, L, K_{AB}, B,\ \{T_S, L, K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3. $A \rightarrow B:\ \{T_S, L, K_{AB}, A\}_{K_{BS}},\ \{A, T_A\}_{K_{AB}}$
4. $B \rightarrow A:\ \{T_A + 1\}_{K_{AB}}$

Here, T_S and T_A are timestamps, and L is a lifetime. Our redesign will have the following presentation.

1. $A \rightarrow S:\ A, B$
2. $S \rightarrow A:\ T_S, L, [\![K_{AB}]\!]_{K_{AS}}, [T_S, L, [\![K_{AB}]\!]_{K_{AS}}, B]_{K_{AS}},$
   $[\![K_{AB}]\!]_{K_{BS}}, [T_S, L, [\![K_{AB}]\!]_{K_{BS}}, A]_{K_{BS}}$
3. $A \rightarrow B:\ T_S, L, [\![K_{AB}]\!]_{K_{BS}}, [T_S, L, [\![K_{AB}]\!]_{K_{BS}}, A]_{K_{BS}}, T_A, [T_A, A]_{K_{AB}}$
4. $B \rightarrow A:\ [T_A + 1, B]_{K_{AB}}$

In this redesign, the implementation of $[\![\cdot]\!]_K$ can use a block cipher in ECB mode. Here we are aware that in this protocol, only the delivered session key K_{AB} is an element that needs protection in terms of confidentiality. Since a session key generated by the trusted server should be new and essentially random, the ECB mode of operation not only suffices to protect its secrecy, but also saves the handling of IV of CBC algorithm. Additional block(s) may be used if the the size of the session key is larger than one block. If the end of the last block has unused bits then these bits should be filled with a random value. Thus, we minimise redundancy in the input data of the ECB algorithm and hence minimise

the possibility for cryptanalysis.

The timestamps, the lifetime and the principal names are no longer treated as secrets. The message origin of these data are protected by means of $[\cdot]_K$. Here $[\cdot]_K$ can be implemented by a suitable hash function or a suitable checksum mechanism. These mechanisms are oneway, or non-reversible and they produce a small amount of ciphertext. The redesigned protocol will not generate any plaintext/ciphertext pair.

Finally, we list two versions of Otway-Rees protocol redesign. Since the specifications are self-evident in capturing the meanings of the protocol, we will omit the explanations regarding implementation issues.

1. $A \rightarrow B:\ M, A, B, N_A, [M, N_A, A, B]_{K_{AS}}$
2. $B \rightarrow S:\ M, A, B, N_A, N_B, [M, N_A, A, B]_{K_{AS}}, [M, N_B, A, B]_{K_{BS}}$
3. $S \rightarrow B:\ M, [\![K_{AB}]\!]_{K_{AS}}, [M, N_A, [\![K_{AB}]\!]_{K_{AS}}]_{K_{AS}},$
   $[\![K_{AB}]\!]_{K_{BS}}, [M, N_B, [\![K_{AB}]\!]_{K_{BS}}]_{K_{BS}}$
4. $B \rightarrow A:\ M, [\![K_{AB}]\!]_{K_{AS}}, [M, N_A, [\![K_{AB}]\!]_{K_{AS}}]_{K_{AS}}$

Considering that the message integrity of the first two lines can be verified by the two client principals in lines 3 and 4 if correct principal names are integrated in the replied messages, this redesign can further be optimised into the following version:

1. $A \rightarrow B:\ M, A, B, N_A$
2. $B \rightarrow S:\ M, A, B, N_A, N_B$
3. $S \rightarrow B:\ M, [\![K_{AB}]\!]_{K_{AS}}, [M, N_A, [\![K_{AB}]\!]_{K_{AS}}, B]_{K_{AS}},$
   $[\![K_{AB}]\!]_{K_{BS}}, [M, N_B, [\![K_{AB}]\!]_{K_{BS}}, A]_{K_{BS}}$
4. $B \rightarrow A:\ M, [\![K_{AB}]\!]_{K_{AS}}, [M, N_A, [\![K_{AB}]\!]_{K_{AS}}, B]_{K_{AS}}$

6 Conclusion

We have reasoned that currently applied methods for authentication protocol specification do not form good mechanisms for protocol development. A new approach is then proposed which consists of an idea of refining the expression of requirements in protocol specifications and a methodology for protocol development. The benefit from taking our new approach to protocol development can be analogous to that from taking the refinement approach to software development. Namely, it allows to precisely express and helps to correctly understand authentication protocols.

Finally we point out that although the protocols investigated in this paper employ symmetric cryptographic techniques, the central idea regarding specification refinement, encryption mode considerations and redundancy usage applies to protocols using asymmetric techniques.

Acknowledgements We would like to thank anonymous referees for helpful comments.

References

[1] C. Boyd and W. Mao. On a limitations of BAN logic. In *Lecture Notes in Computer Science 765*, pages 240–247. Springer-Verlag, 1993.

[2] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. Technical Report SRC Technical Report 39, Digital Equipment Corporation, February 1989.

[3] ISO/IEC. 10116: Information processing - modes of operation for an n-bit block cipher algorithm, 1991.

[4] ISO/IEC. CD 9798-1, information technology - security techniques - entity authentication mechanisms - part 1: General model, 1991-09-01.

[5] ISO/IEC. N 739, DIS 9798-2, information technology - security techniques - entity authentication mechanisms - part 2: Entity authentication using symmetric techniques, 1993-08-13.

[6] ISO/IEC. CD 11770-2: Key management, part 2: Key management mechanisms using symmetric techniques, 1993-10-03.

[7] Kohl J. and C. Neuman. The Kerberos network authentication service (v5). Internet Archive RFC 1510, September 1993.

[8] S.P. Miller, C. Neuman, J.I. Schiller, and J.H. Saltzer. Kerberos authentication and authorization system. Project Athena Technical Plan Section E.2.1, 1987.

[9] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *CACM*, 21(12):993–999, 1978.

[10] D. Otway and O. Rees. Efficient and timely mutual authentication. *Operating Systems Review*, Vol 21(1):8–10, 1987.

[11] S.G. Stubblebine and V.D. Gligor. On message integrity in cryptographic protocols. In *1992 IEEE Symposium on Security and Privacy*, pages 85–104. IEEE Computer Society Press, 1992.