Parsing ambiguities in authentication and key establishment protocols

Liqun Chen
Hewlett-Packard Laboratories
Filton Road
Stoke Gifford
Bristol BS34 8QZ, UK
liqun.chen@hp.com

Chris J. Mitchell
Royal Holloway
University of London
Egham
Surrey TW20 0EX, UK
c.mitchell@rhul.ac.uk

30th September 2008

Abstract

A new class of attacks against authentication and authenticated key establishment protocols is described, which we call *parsing ambiguity attacks*. If appropriate precautions are not deployed, these attacks apply to a very wide range of such protocols, including those specified in a number of international standards. Three example attacks are described in detail, and possible generalisations are also outlined. Finally, possible countermeasures are given, as are recommendations for modifications to the relevant standards.

1 Introduction

Over the last four years a number of new attacks have been published on long-established and apparently stable standardised authenticated key establishment protocols. The origin of these protocols can be traced back to the seminal paper of Needham and Schroeder [24], and the protocols concerned had been widely studied and were believed to be secure. Indeed, the first edition of the international standard for key establishment mechanisms using symmetric cryptography, ISO/IEC 11770-2, appeared in 1996 [8], and no problems were identified until 2004.

However, things have changed in recent years, with the publication of a number of attacks (including a range of 'type attacks') on two standardised protocols. The attacked protocols (mechanisms 12 and 13 of ISO/IEC 11770-2) both assume that the two parties who wish to establish a shared secret key already share a secret key with a trusted third party (acting as a key translation centre).

- In 2004 Cheng and Comley [5, 6] described two attacks on mechanism 12 from ISO/IEC 11770-2:1996 [8], one a replay attack and the other a type attack.
- This caused ISO/IEC SC27, the committee responsible for developing and maintaining the standard, to issue a corrigendum [14] withdrawing the mechanism concerned.
- Shortly afterwards, work started within SC27 on a new version of the standard containing a replacement 'fixed' version of mechanism 12; the revised text was published in 2008 [15] as the 2nd edition of ISO/IEC 11770-2.
- In mid 2008, Mathuria and Sriram [19] described new type attacks on both mechanism 13 and the 'fixed' version of mechanism 12 proposed by Cheng and Comley [5, 6]. Whilst the attack on mechanism 13 will need to be addressed by SC27, the attack on the Cheng-Comley version of mechanism 12 is not an immediate concern; this is because the version of mechanism 12 contained in [15] is different to the Cheng-Comley fix, and is not subject to the Mathuria-Sriram attacks.

The term 'type attack' refers to a protocol attack in which protocol fields of one 'type' are misrepresented as being fields of a different type. This typically means that all or part of a message legitimately sent in step i of a protocol, is misrepresented as a message sent in step j ($j \neq i$) of a separate instance of the protocol. For a helpful review of the various types of attacks possible on authentication and key establishment protocols, see [4]. Before proceeding we note that the Mathuria-Sriram attacks require three consecutive fields in one protocol message to be misinterpreted as a single field in a different protocol message. Thus in some ways these attacks are the first step towards the much more general class of attacks described in this paper, since they require the protocol to be used in a domain in which protocol fields can have varying lengths.

In this paper we show that a wide range of attacks are possible against almost all the standardised protocols. The attacks we describe are related to type attacks, but are distinct in that they do not require a message sent at one step of a protocol to be (mis)used at a distinct protocol step. We call these attacks *parsing ambiguity attacks* since they require strings of data input to cryptographic algorithms to be capable of being interpreted in more than one way. In some ways they can be regarded as specification flaws rather than flaws in the protocols themselves; nevertheless, the same problem arises in the specifications of many protocols.

We remark also that, although it is true that some of the protocols we attack do not have proofs of security (although some do, e.g. mechanism 9 of ISO/IEC 11770-2 [15] is essentially identical to the three party key distribution protocol proved secure by Bellare and Rogaway in [2]), the method of attack is essentially independent of this fact. That is, these attacks fall outside the scope of current security proofs, and hence any provably secure protocol specified in the same way is likely to be subject to the attacks we describe. This is because protocol security proofs typically do not consider the possibility of one sequence of data items subjected to a cryptographic function being misinterpreted as a different sequence of data items — that is, correct/unique parsing is either an implicit or an explicit assumption.

2 Some simple examples

In this section we give three examples of parsing ambiguity attacks to illustrate the general approach. All these attacks apply to well-established standardised protocols, using a range of cryptographic techniques. We also discuss how these attacks might be realised in practice. In the next section we generalise these examples, to show the broad applicability of the attack technique.

2.1 Specifying authentication and authenticated key establishment protocols

We start by considering how the protocols of concern to us here are typically specified. It is this method of specification that gives rise to the possibility of attacks.

A variety of terminologies have been used to specify authentication and key establishment protocols. In this paper we use the terminology adopted in the ISO/IEC 9798 and 11770 series of standards, covering authentication protocols and key establishment mechanisms respectively. To show the use of this terminology we give an example of a mechanism from the current version of ISO/IEC 11770-2 [15], specifically mechanism 8. This mechanism is essentially identical to the basic *Kerberos* protocol [18, 25] as specified in [4].

This protocol involves two parties A and B, who wish to share a secret session key. A and B both share a secret session key with a trusted third party P, where the key shared by A and P is denoted by K_{AP} and the key shared by B and P is denoted by K_{BP}. A and B are assumed to know with whom they are trying to establish a key, and to know what their respective identifiers are.

The mechanism involves the following message flows:

(1): $A \rightarrow P$: $TVP_A||I_B$;

(2): $P \rightarrow A$: $e_{K_{AP}}(TVP_A||F||I_B||\text{Text}_1)$ || $e_{K_{BP}}(T_P/N_P||F||I_A||\text{Text}_2)$;

(3): $A \rightarrow B$: $e_{K_{BP}}(T_P/N_P||F||I_A||\text{Text}_2)$ || $[e_K(T_A/N_A||I_B||\text{Text}_3)]$;

(4): $B \to A$: $[e_K(T_B/N_B||I_A||\text{Text}_4)]$.

In this description, $X \to Y$: Z indicates that entity X sends message Z to entity Y, I_X denotes an identifier for entity X (where I_X is assumed to be unique within the domain of application of the protocol), $e_{\hat{K}}(Z)$ denotes the authenticated encryption of data Z using secret key $\hat{K}$ (using an authenticated encryption mode such as EAX [3] or GCM [23] — see also ISO/IEC 19772 [17]), TVP_X represents a time variant parameter (TVP) issued by entity X, i.e. a random number R_X, time stamp T_X or sequence number N_X, F represents keying material, and Text_i ($i = 1, 2, \ldots$) are optional fields that can be used to carry application-specific data.

All data items within the various messages are (implicitly) assumed to be bit strings, and $||$ represents the bit concatenation operator (note that this assumption about the interpretation of $||$ is explicitly given in all the international standards discussed in this paper). Data items shown in square brackets are optional.

In message (1), A requests keying material from P by sending an identifier I_B for B together with a TVP. In (2), P sends A newly generated keying material F (containing a key K) together with the TVP sent by A in (1) and an (optional) text field. Message (2) contains two main parts, i.e. $e_{K_{AP}}(TVP_A||F||I_B||\text{Text}_1)$ and $e_{K_{BP}}(T_P/N_P||F||I_A||\text{Text}_2)$. On receipt of (2), A deciphers the first part, checks the correctness of the TVP and the identifier I_B, and obtains the key K from F. In message (3), A forwards the second part of message (2) to B, optionally including the second part $[e_K(T_A/N_A||I_B||\text{Text}_3)]$, which, if present, enables B to check the integrity of the key K retrieved from F. On receipt of (3), B deciphers the encrypted part, checks the correctness of T_P or N_P and the identifier I_A, and obtains the key K from F. Optionally, B sends message (4) to A, thereby acknowledging that it shares the key K with A.

A somewhat distinct terminology is often employed — see, for example, Boyd and Mathuria [4]. In this terminology, the above protocol would be specified as follows:

(1): $A \to P$: TVP_A, I_B;

(2): $P \to A$: $\{TVP_A, F, I_B, \text{Text}_1\}_{K_{AP}}$, $\{T_P/N_P, F, I_A, \text{Text}_2\}_{K_{BP}}$;

(3): $A \to B$: $\{T_P/N_P, F, I_A, \text{Text}_2\}_{K_{BP}}$,$[\{T_A/N_A, I_B, \text{Text}_3\}_K]$;

(4): $B \to A$: $[\{T_B/N_B, I_A, \text{Text}_4\}_K]$.

That is, $\{Z\}_{\hat{K}}$ is used to denote the encryption of Z using the key $\hat{K}$, and the comma functions as a concatenation operator. Although distinct in style, the notation has a similar interpretation. Sometimes (as in [4]) the meaning of the comma is not made explicit, and hence the interpretation of X, Y could vary anywhere between the simple concatenation of X and

Y to the encoding of fields X and Y using a scheme such as XML or one of the encoding rules associated with ASN.1. However, in the absence of specific advice to the contrary, we make the reasonable assumption that some implementers will interpret the comma as simple concatenation.

2.2 An attack on a key distribution protocol

The first attack we describe applies to mechanism 8 from ISO/IEC 11770-2 [15], described immediately above. In this attack, party C impersonates party A (or B) to B (or A) in the key establishment process, as a result of which B (or A) will possess a secret key which B (or A) believes is shared with A (or B), but which is, in fact, known to C. We also suppose that C is a legitimate member of the domain, i.e. it shares a secret key K_{CP} with P, and has the right to ask P to set up a key with A or B. Three possible variants of the attack are described, each based on different assumptions about the nature of identifier fields, keying material, text fields, and nonces.

We first note that the protocol specification does not restrict the lengths of the keying material and identifier fields, and we take advantage of this fact in all three variants of the attack. In all cases we also suppose that C is able to choose its own identifier.

In order to conduct the first variant of the attack we suppose that C chooses $I_C = 0||I_A$, i.e. I_C is one bit longer than I_A. The attack operates as follows. Here and throughout we write $X(Y)$ to mean that entity X is performing an action whilst pretending to be entity Y.

(1): $C \rightarrow P$: $TVP_C||I_B$;

(2): $P \rightarrow C$: $e_{K_{CP}}(TVP_C||F||I_B||\text{Text}_1)$ || $e_{K_{BP}}(T_P/N_P||F||I_C||\text{Text}_2)$;

(3): $C(A) \rightarrow B$: $e_{K_{BP}}(T_P/N_P||F||I_C||\text{Text}_2)$ || $[e_K(T_{C(A)}/N_{C(A)}||I_B||\text{Text}_3)]$;

(4): $B \rightarrow C(A)$: $[e_K(T_B/N_B||I_A||\text{Text}_4)]$.

Observe that messages (1) and (2) are the perfectly legitimate first two messages of the protocol, as they would be used by C when wishing to establish a shared key with B. However, because of the special form of C's identifier, C is able to misuse the encrypted string provided by P in message (2) in order to impersonate A to B in message (3). To achieve this goal, at the same time as sending message (3) to B, C asserts to B that it is A, and that the bit-length of the keying material field is one greater than the bit-length of the field F generated by P.

To see what effect this has, observe that $F||I_C = F||0||I_A = F^*||I_A$, where F^* contains one more bit than F. That is, since B expects the keying material field in message (3) to be of length one greater than the length of F, B will decrypt message (3) to obtain $T_P/N_P||F^*||I_A||\text{Text}_2$, which will look exactly as it should. That is, B will believe F^* is shared with A whereas it is actually shared with C.

If changing the length of F is not possible, e.g. because it is fixed to a single value for all members of the domain, then a similar result is possible by manipulating the length of the Text_2 field. That is, suppose C chooses his/her identifier so that $I_C = I_A||0$. The second attack variant is then precisely as above, except that C asserts to B that it is A, and that the bit-length of the Text_2 field is one greater than the bit-length of the field Text_2 generated by P. B will decrypt message (3) to obtain $T_P/N_P||F||I_A||\text{Text}_2^*$, where $\text{Text}_2^* = 0||\text{Text}_2$, and will believe it shares F with A, whereas it will actually share it with C.

Observe that the attack may be simpler to launch in this second case, since the exact form of the text field Text_2 may not be prescribed by the application. Indeed, recipients may choose to ignore text fields if they are not expecting them to contain any useful information. In such a case, varying the length of this field may not cause any concern to the recipient.

Next, suppose that the implementation is such that the length of neither the keying material field F nor the text fields can be changed. To describe our third attack variant we observe that the lengths of random nonces used in the above protocol have not been specified in ISO/IEC 11770-2 [15]. If a recipient (e.g. entity P) is prepared to accept varying length nonces, then a variant of the attack is still possible. For example, suppose the attacker C chooses as its identifier $I_C = 0||I_B$ for one of the victims B. The attack operates as follows.

(1): $A \to C(P)$: $R_A||I_B$;

(1'): $C(A) \to P$: $R'_A||I_C$;

(2): $P \to A$: $e_{K_{AP}}(R'_A||F||I_C||\text{Text}_1)$ || $e_{K_{CP}}(T_P/N_P||F||I_A||\text{Text}_2)$;

(3): $A \to C(B)$: $e_{K_{CP}}(T_P/N_P||F||I_A||\text{Text}_2)$ || $[e_K(T_A/N_A||I_B||\text{Text}_3)]$;

(4): $C(B) \to A$: $[e_K(T_{C(B)}/N_{C(B)}||I_A||\text{Text}_4)]$.

C first intercepts message (1) from A to P, and prevents it from reaching P. C next removes the last bit of R_A to obtain R'_A, i.e. $R_A = R'_A||x$, for some bit x. C (masquerading as A) then sends message $(1')$ to P to persuade P that A wants to set up a key with C. Note that message $(1')$ is equal to $R'_A||0||I_B$. Message (2) is the legitimate message from P to A. After decrypting the first part of the message, A obtains $R'_A||F||I_C||\text{Text}_1$, and interprets it as $(R'_A||y)||(F'||0)||I_B||\text{Text}_1$, where $F = y||F'$ for some bit y. If $y = x$ (which holds with probability 1/2), A will accept $F^* = F'||0$ as valid keying material, which A believes to be shared with B; otherwise A will reject the message and possibly restart the protocol. If the first part of message (2) is accepted, A will forward the second part to C (impersonating

B) in message (3). Obviously, C is able to decrypt the first part to obtain F^*. Observe that the two data items in square brackets are perfectly legitimate parts of the protocol, and will not help A to detect the attack.

Note that essentially the same attack works on mechanism 9 of ISO/IEC 11770-2 [15] in an even more powerful way. In this case, by manipulating the nonce R_A and its identifier I_C, the attacker C can impersonate B to A; by manipulating the nonce R_B and its identifier I_C, the attacker C can also impersonate A to B.

Note also that the above attacks can be generalised to cover the cases where either (a) the identifier of I_C is a number of bits longer than I_A (or I_B), and/or I_C is slightly shorter than I_A (or I_B). In summary, if C can choose its own identifier, and C is allowed to 'modify' the length of any of F, Text_2 or R_A, C can successfully attack the protocol.

2.3 A parsing-based reflection attack

The next attack we describe applies to a somewhat simpler protocol, namely the one-pass authentication mechanism of ISO/IEC 9798-2 [12]. It works on the assumption that A and B already share a long-term secret key K_{AB}, and that they wish to use this to provide (unilateral) authentication of A to B. The mechanism involves the following single message flow:

(1): $A \to B$: $e_{K_{AB}}(T_A/N_A||I_B||\mathrm{Text}_1)$.

In message (1), A sends B either a time stamp T_A or a sequence number N_A, the identifier I_B, and an optional text field Text_1. The data fields are encrypted using the key K_{AB}, where, as previously, use of an authenticated encryption mechanism is assumed. On receipt of the message, B deciphers the encrypted part and checks I_B and the time stamp or sequence number. If the checks succeed, B deems A to have been authenticated.

As a result of the attack, C manages to persuade B that it has authenticated A, whereas B is actually communicating with C. The attack has similarities to the reflection attack described in [20]. In order to make the attack work, we suppose that B has been issued with an identifier I_B with the property that $I_A = I_B||x$, where x is a short bit-string. We also suppose that time-stamps are used, and not sequence numbers.

C starts by impersonating A to B. When B asks C for an authentication message, C immediately starts a second communications session with B again pretending to be A, this time asking B to authenticate itself to A. Following the protocol above, B will send C the following:

$$e_{K_{AB}}(T_B||I_A||\mathrm{Text}_1).$$

Now, since $I_A = I_B||x$, we know that $I_A||\mathrm{Text}_1 = I_B||\mathrm{Text}_1^*$, where $\mathrm{Text}_1^* = x||\mathrm{Text}_1$.

On receipt of the above message from B, C immediately sends precisely the same message back to B as the response to B's initial request for authentication. When B decrypts it, B will obtain the string $T_B||I_B||\text{Text}_1^*$. Assuming that the message is 'reflected' back to B quickly enough, B will accept T_B as current, and will also recognise I_B. As long as the optional data field Text_1^* is also acceptable (and whether or not this is true will be very much application specific), B will accept the message as having come from A, and will therefore have (falsely) verified the presence of A.

An almost identical attack applies to the corresponding MAC-based protocol from ISO/IEC 9798-4 [11]. This protocol involves the following single message:

(1): $A \rightarrow B$: $T_A/N_A||\text{Text}_2||f_{K_{AB}}(T_A/N_A||I_B||\text{Text}_1)$.

where f is a MAC function (such as a CBC-MAC [10] or HMAC [13]), and other notation is as above.

In this case the attack may actually be easier to launch, as the data string used to compute the MAC is reassembled by the recipient from the information provided by the sender (as opposed to being recovered as the result of a decryption operation). That is, the issue is not whether or not the recipient can be persuaded to parse a decrypted string incorrectly, but whether or not the recipient can be persuaded that a certain sequence of data fields, as provided in cleartext by the sender, are a valid sequence of data fields. In this latter case it seems reasonable to suppose that the recipient could be flexible about the lengths of some of the fields involved.

2.4 An attack on a signature-based authentication protocol

Our third example attack applies to an authentication protocol based on the use of digital signatures, namely the two-pass mutual authentication protocol from ISO/IEC 9798-3 [9]. This protocol has the following two message flows:

(1): $A \rightarrow B$: $T_A/N_A||I_B||\text{Text}_2||s_A(T_A/N_A||I_B||\text{Text}_1)$;

(2): $B \rightarrow A$: $T_B/N_B||I_A||\text{Text}_4||s_B(T_B/N_B||I_A||\text{Text}_3)$.

In this description, s_X denotes the signature function of entity X, and other notation is as above. Note that if necessary, one or both messages can include a copy of the certificate for the sender's signature verification public key. Note also that Text_2 must contain sufficient information to enable B to construct Text_1 — a typical case would be to put $\text{Text}_2 = \text{Text}_1$, and this is what we assume below (precisely corresponding remarks apply to Text_3 and Text_4).

One possible attack on this scheme operates as follows. We suppose that the domain within which the protocol is used employs timestamps rather than sequence numbers. Entity C (which must be a legitimate party) registers an identifier $I_C = I_B||x$ for some (short) bit-string x. C now initiates an instance of the above protocol with A using identity I_C, so that we have:

(1): $C \rightarrow A$: $T_C||I_A||\text{Text}_1||s_C(T_C||I_A||\text{Text}_1)$;

(2): $A \rightarrow C$: $T_A||I_C||\text{Text}_3||s_A(T_A||I_C||\text{Text}_3)$.

C now re-uses an unchanged copy of message (2) to initiate a second instance of the protocol with B, this time impersonating A. B then responds in the expected way. That is we have:

(1): $C(A) \rightarrow B$: $T_A||I_B||\text{Text}_3^*||s_A(T_A||I_B||\text{Text}_3^*)$;

(2): $B \rightarrow C(A)$: $T_B||I_A||\text{Text}_3'||s_B(T_B||I_A||\text{Text}_3')$;

where $\text{Text}_3^* = x||\text{Text}_3$. This works because $I_C||\text{Text}_3 = I_B||x||\text{Text}_3 = I_B||\text{Text}_3^*$. Thus the attack will be successful if B accepts $\text{Text}_3^* = x||\text{Text}_3$ as a valid text field.

This is an attack since A believes it has completed a mutual authentication with C, whereas B believes it has conducted a mutual authentication with A.

2.5 Realising the attacks

We now describe possible ways in which the attacks might be realised in practice; that is we describe why they are a genuine threat (at least in some application scenarios).

First observe that all the above attacks require at least two of the protocol message fields to be of variable length. We now consider some examples of cases where this might be a reasonable assumption. We consider each type of message field in turn.

- *Keying material fields F*: In some cases, more than one type of key might be distributed within a domain. In such a case, the keys involved may be of varying length.

- *Identifier fields I_X*: In practice, whilst many types of identifier have fixed length, there are examples of identifiers which do not have this property. Examples of widely used identifiers of varying length include email addresses and URLs.

- *Optional text fields Text_i*: The nature of these fields will be application dependent, and it seems reasonable to assume that in some applications these can be of variable length.

- *Nonces* R_X: Nonces are used to prevent replay or interleaving attacks. For this purpose, a nonce must be unpredictable and non-repeating with a high probability. Given that there may be varying assessments of what is a suitably high probability, users may choose varying lengths for nonces.

We also require that it is possible for two identifiers (I_A, I_B for parties A and B, say) to have the property that $I_A = x||I_B$, for some x. Depending on the nature of the attack, we either require it to be possible for A:

- given I_B, to choose I_A such that $I_A = x||I_B$ (or $I_A = I_B||x$), or
- to find a pair of entities A and B with the property that $I_A = x||I_B$.

Email addresses can satisfy both these properties. In the first case, it may be relatively straightforward to register an email address which has another (specified) email address as a substring. For example, suppose B has email address `b.smith@rhul.ac.uk`. Then A could register the email address `bob.smith@rhul.ac.uk`, i.e. $I_A = \texttt{bo}||I_B$.

The second case can arise commonly, e.g. in an organisation where the first part of the address is a name. The example given above could easily arise by chance. Indeed, we expect that there will be several examples of such pairs of addresses in any large organisation which allows email addresses to be based on the names of individuals.

3 Widening the attack scope

In section 2 we described three examples of 'parsing attacks' against a range of different types of authentication and authenticated key establishment protocols. In fact, it would seem possible to launch similar attacks against almost any authentication or authenticated key establishment protocol, at least if they are specified in the style given above. Of course, if the lengths of all the fields are fixed throughout the domain of use then the attacks are prevented, but the imposition of such a requirement is rare — also, it is simply not appropriate to make such a requirement for certain types of identifier (as discussed above).

To see how the more general classes of attack apply, we make the following observations.

- The approach used in the attack described in section 2.2 can also be applied to attack any of the seven third-party-based authenticated key establishment mechanisms in ISO/IEC 11770-2:2008 [15], including both Key Distribution Centre and Key Translation Centre based mechanisms. The attack approach also applies to the two third-party-based authentication protocols given in ISO/IEC 9798-2 [12].

- The reflection attack described in section 2.3 also applies to other standardised authentication protocols based on symmetric cryptography, including both those based on nonces (i.e. challenge-response protocols) as well as protocols providing mutual authentication of two parties. That is, the attack applies to the majority of the protocols specified in ISO/IEC 9798-2 [12] and ISO/IEC 9798-4 [11].

- The attack described in section 2.4 could be generalised in a number of ways. For example, as noted above, the lengths of random nonces used in some standardised protocols have not been specified. In protocols using nonces, the nonce fields could also be used to 'swap' identifiers. For example, adopting such an approach, the three pass authentication mechanism in ISO/IEC 9798-3 [9] can be attacked in a similar way to the attack given in section 2.4.

- Additional problems of the same general type can arise with the public key based key establishment mechanisms specified in ISO/IEC 11770-3 [16], when identity-based mechanism are used. For example, in public key transport mechanism 1 of ISO/IEC 11770-3, $\text{PKI}_A||\text{Text}$ is distributed over an authentic channel, where PKI_A contains the public key of entity A. If an application is using identity-based cryptography, PKI_A will contain an identity-based public key, which includes the public key of a key generation centre (KGC) and an identifier I_A for A. Now suppose that an attacker C registers with the same KGC as A, and uses an identifier I_C which includes I_A as a substring. In such a case, it may be possible for C to persuade the recipient of $\text{PKI}_A||\text{Text}$ to incorrectly parse it. When this happens, the recipient will accept the identity I_A with a public key for which C knows the corresponding private key.

- Finally observe that two of the protocols given in ISO/IEC 9798-3 have also been adopted in NIST FIPS Pub. 196 [22]. Exactly corresponding notation and assumptions have been used in the NIST document, and hence the attacks applying to the ISO/IEC protocols also apply to the protocols in this standard.

In conclusion, we observe that the attacks we have described are probably just the tip of an iceberg. We certainly do not believe that we have discovered all the possible classes of attack that can arise because of ambiguities in the interpretation of cryptographically-protected data strings.

4 Fixing the problem

It should be clear that the origin of all the problems we have described is the possible false interpretation of a cryptographically protected string. We have identified two main types of 'parsing error':

1. where the attacker persuades the recipient of an encrypted string to parse the decrypted string incorrectly;

2. where the attacker persuades the recipient of a MAC (or signature) to believe that the MAC (or signature) has been computed on a string made up of the combination of a series of data fields which differ from those used to compute the MAC (or signature).

Observe that these two types of problem are very general, and have been studied in a variety of different contexts. That is, the problems we have identified here might arise in contexts other than authentication or key establishment; issues arising from situations in which a signer is required to sign data strings provided by other (potentially malicious) parties have been explored previously in [21]. Also, if a malicious party A could manage to have his/her signature $s_A(m_1||m_2)$ interpreted as $s_A(m_3||m_4)$, he/she might be able to escape from a non-repudiation commitment.

Problems of these two types can be avoided if the parties in any domain agree in advance on a way of:

1. composing data strings that are to be encrypted in such a way that they can be parsed unambiguously;

2. combining data strings to generate a bit string prior to computing a MAC (or signature) in such a way that only precisely the same sequence of data strings can yield the bit string.

This could be achieved in a variety of different ways. For example, both properties could be guaranteed by:

- fixing the lengths of each of the substrings throughout the domain (this would mean that inherently variable length identifiers such as email addresses would need to be padded to a pre-agreed fixed length);

- encoding each substring in a way that guarantees uniqueness, e.g. using a tag-length-value format or XML.

It is interesting to note that, in two of the key papers establishing the 'provable security' approach to authentication and authenticated key establishment protocols, Bellare and Rogaway [1, 2] carefully specified 'a random k-bit challenge' for a nonce and 'k-bit names of the players' for identities. It would seem likely that, without fixing the length of each parameter, mechanisms such as those attacked here cannot be proved secure. We next give an example to demonstrate this.

As stated before, the three party key distribution protocol proposed in [2] is essentially identical to mechanism 9 of ISO/IEC 11770-2 [15]. The security of this protocol was proved in [2] in the security model proposed in the same paper. We now remove the condition that both the nonces and identifiers are of k bits, and allow an attacker C to manipulate their lengths. Suppose C is able to persuade the key distribution server S to accept different lengths for R_A and I_C. By following the same approach as described in Section 2.2, C removes the first bit (x, say) of R_A to obtain R'_A, and arranges for its identifier I_C to be $I_C = I_B||x$. Since $I_B||R_A = I_C||R'_A$, C can successfully impersonate party B to party A.

We now attempt to verify the security of this slightly modified protocol in the security model presented in [2], where the details of the model are given in [2]. C is able to share the established key with A but C is not the partner of A, since A is attempting to share the key with B. An adversary in the security model with the goal of breaking this protocol can take an oracle $\Pi^s_{A,B}$, which models an instance s of party A attempting to agree a shared session key with party B, as a Test oracle, and then make a Reveal query to another oracle $\Pi^t_{C,A}$, which similarly models an instance t of party C attempting to agree a key with party A. Because C is not A's partner, the session key held by C should be revealed to the adversary. So the adversary can easily win the game that specifies the model, and the protocol is therefore not secure in this model.

Another possible approach to addressing the identified problems would be to replace I_X with $h(I_X)$ in every cryptographically protected string for every identifier I_X, where h is a cryptographic hash function. This would help to prevent any attacks where one identifier is confused for another.

Finally, where possible it may also be prudent to add message identifiers to each cryptographically data string, as has been done for mechanism 12 in the latest edition of ISO/IEC 11770-2 [15]. Whilst this does not directly relate to the attacks described here, it prevents attacks where one message of a protocol is abused as a different message of the same protocol.

5 Implications for protocol standards

The implications for the ISO/IEC standards covering authentication and authenticated key establishment appear to be clear. The current specifications are not sufficient to ensure that every implementer of the standards will produce a secure scheme. That is, additional guidance needs to be provided to users of the standards on the construction and interpretation of cryptographically protected data strings. To avoid problems we cannot detect right now, it would be a good idea to requirements along the lines of those given in the previous section to all parts of ISO/IEC 9798 and ISO/IEC 11770.

Finally, although the main focus of this paper has been on protocols specified in international standards, similar problems arise in many places that protocols are specified. This includes academic papers and books, such as that of Boyd and Mathuria [4]. All users of authentication and authenticated key establishment protocols should be aware of the possible issues arising from parsing ambiguities.

6 Concluding remarks

From a general perspective, the problems identified here are just one example of issues that can arise when translating theory into practice. Theorems about protocols (e.g. established using one of many logic-based approaches or the 'provable security' techniques) are built upon important assumptions without which the proofs do not hold. These assumptions may be explicit (as in the assumptions about fixed-length strings made by Bellare and Rogaway), or implicit (as in some of the logic-based approaches, where the unique division of an encrypted string into sub-fields is assumed).

This highlights the importance of correctly bridging the gap between theory and practice. We conclude by observing that the attacks described here are by no means the only examples of attacks of this general type — see, for example, the recent work of Degabriele and Paterson [7] on attacking IPsec used in encryption-only configurations.

References

[1] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *Advances in Cryptology — CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22–26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, Berlin, 1993.

[2] M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May–1 June 1995, Las Vegas, Nevada, USA*, pages 57–66. ACM Press, 1995.

[3] M. Bellare, P. Rogaway, and D. Wagner. The EAX mode of operation. In B. Roy and W. Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer-Verlag, Berlin, 2004.

[4] C. A. Boyd and A. Mathuria. *Protocols for key establishment and authentication.* Springer-Verlag, 2003.

[5] Z. Cheng and R. Comley. Attacks on an ISO/IEC 11770-2 key establishment protocol. Cryptology ePrint Archive: Report 2004/249, September 2004.

[6] Z. Cheng and R. Comley. Attacks on an ISO/IEC 11770-2 key establishment protocol. *International Journal of Network Security*, 3:290–295, 2006.

[7] J. P. Degabriele and K. G. Paterson. Attacking the IPsec standards in encryption-only configurations. In *Proceedings: 2007 IEEE Symposium on Security and Privacy (S&P 2007), 20–23 May 2007, Oakland, California, USA*, pages 335–349. IEEE Computer Society Press, Los Alamitos, California, 2007.

[8] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 11770-2:1996, Information technology — Security techniques — Key management — Part 2: Mechanisms using symmetric techniques*, 1996.

[9] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 9798-3:1998, Information technology — Security techniques — Entity authentication — Part 3: Mechanisms using digital signature algorithms*, 2nd edition, 1998.

[10] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 9797–1, Information technology — Security techniques — Message Authentication Codes (MACs) — Part 1: Mechanisms using a block cipher*, 1999.

[11] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 9798–4: 1999, Information technology — Security techniques — Entity authentication — Part 4: Mechanisms using a cryptographic check function*, 2nd edition, 1999.

[12] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 9798-2:1999, Information technology — Security techniques — Entity authentication — Part 2: Mechanisms using symmetric emccipherment algorithms*, 2nd edition, 1999.

[13] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 9797–2, Information technology — Security techniques — Message Authentication Codes (MACs) — Part 2: Mechanisms using a hash-function*, 2000.

[14] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 11770-2:1996/Cor.1:2005, Information technology — Security techniques — Key management — Part 2: Mechanisms using symmetric techniques: Technical Corrigendum 1*, April 2005.

[15] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 11770-2:2008, Information technology — Security techniques — Key management — Part 2: Mechanisms using symmetric techniques*, 2nd edition, 2008.

[16] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 11770-3:2008, Information technology—Security techniques—Key management; Part 3: Mechanisms using asymmetric techniques*, 2nd edition, 2008.

[17] International Organization for Standardization, Genève, Switzerland. *ISO/IEC FDIS 19772, Information technology — Security techniques — Authenticated encryption mechanisms*, July 2008.

[18] J. Kohl and C. Neuman. *RFC 1510, The Kerberos Network Authentication Service (V5).* Internet Engineering Task Force, September 1993.

[19] A. Mathuria and G. Sriram. New attacks on ISO key establishment protocols. Cryptology ePrint Archive: Report 2008/336, July 2008.

[20] C. J. Mitchell. Limitations of challenge-response entity authentication. *Electronics Letters*, **25**:1195–1196, 1989.

[21] C. J. Mitchell and A. Thomas. Standardising authentication protocols based on public key techniques. *Journal of Computer Security*, 2:23–36, 1993.

[22] National Institute of Standards and Technology (NIST), Gaithersburg, MD. *Federal Information Processing Standards Publication 196 (FIPS PUB 196): Entity Authentication Using Public Key Cryptography*, February 1997.

[23] National Institute of Standards and Technology (NIST). *NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, November 2007.

[24] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, **21**:993–999, 1978.

[25] B. C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, September 1994.