Noname manuscript No.
(will be inserted by the editor)

Improving the ISO/IEC 11770 standard for key management techniques

Cas Cremers and Marko Horvat

the date of receipt and acceptance should be inserted later

Abstract We provide the first systematic analysis of the ISO/IEC 11770 standard for key management techniques [19,20], which describes a set of key establishment, key agreement, and key transport protocols. We analyse the claimed security properties, as well as additional modern requirements on key management protocols, for over 30 protocols and their variants. Our formal, tool-supported analysis of the protocols uncovers several incorrect claims in the standard. We provide concrete suggestions for improving the standard.

Keywords formal analysis, ISO, protocol standards, security protocols

1 Introduction

The International Organisation for Standardisation (ISO) develops and promotes international standards, which include a wide variety of security mechanisms. Many large vendors aim to support ISO standards, for example because they are mandated by oversight bodies [16] or to prevent trade barriers. Hence, it is critical that the ISO standards for security mechanisms are thoroughly scrutinised. However, most previous analyses of the ISO security standards have been very limited in scope, e.g. [10, 11, 17, 25, 27]. One exception is the analysis of Basin et al. of the ISO/IEC 9798 standard for entity authentication [4] in 2012. Their analysis uncovered a series of issues that led to an updated version of the 9798 standard.

In this paper, we focus on the ISO/IEC 11770 standard for key management protocols, in particular on Parts 2 and 3 of this standard. In the most recent version as of June 2014, these two parts together describe 33 base protocols for key establishment, key agreement, and key transport. Many of the standard's protocols are based on protocols such as Diffie-Hellman, variants of MQV, and the TLS handshake. For many of the protocols, at least two variants are described. Thus, analysing these two parts is a significant undertaking.

In positive contrast to other security protocol standards [5], the ISO/IEC 11770 standard explicitly specifies security properties for each of its protocols. Two of these properties are structural properties, i.e. key control and replay detection. Additionally, there are four security properties that relate to active adversaries, namely key authentication, key confirmation, entity authentication, and forward secrecy.

We use tool-supported formal methods to determine if the protocols indeed satisfy their claimed non-structural security properties. We also consider other modern key exchange security properties, such as resilience against Key Compromise Impersonation (KCI) and Unknown Key Share (UKS) attacks.

Contributions We perform the first comprehensive analysis of Parts 2 and 3 of the ISO/IEC 11770 standard. Our analysis uncovers multiple previously unreported errors and weaknesses. For each of the discovered issues, we provide concrete recommendations for improving the standard. Many of the discovered issues could have been prevented if the recent recommendations for related standards, in particular ISO/IEC 9798, had been applied to their counterparts in ISO/IEC 11770.

Our protocol models and tools used are available for download from `http://www.cs.ox.ac.uk/people/cas.cremers/scyther/iso11770/`.

University of Oxford
Oxford, UK

Acknowledgements This work builds on, and extends abstract protocol models originally developed for an earlier analysis of ISO/IEC 11770 by Lara Schmid [28].

Overview In Section 2, we give some background on ISO/IEC 11770 and illustrate some of its protocols. We describe our analysis approach in Section 3 and present the results in Section 4. We provide concrete recommendations for improving the standard in Section 5, discuss related work in Section 6, and conclude in Section 7.

2 Background on ISO/IEC 11770

The ISO/IEC 11770 standard describes key management techniques. According to the standard, the purpose of key management is to provide procedures for handling cryptographic keying material to be used in symmetric or asymmetric mechanisms. Effectively, the standard describes a large number of key agreement, key transport, and key establishment protocols. We will therefore use the terms *mechanism* and *protocol* interchangeably.

The standard is currently divided into five parts. Part 1 was originally released in 1996 and has been updated over the years. It describes the context and framework. Parts 2 and 3 describe mechanisms based on symmetric and asymmetric techniques. Part 4 describes mechanisms based on weak secrets, such as password-based key exchange protocols. Part 5 describes group key management mechanisms. The standard is expected to be extended with a sixth part on key derivation functions.

2.1 Protocols

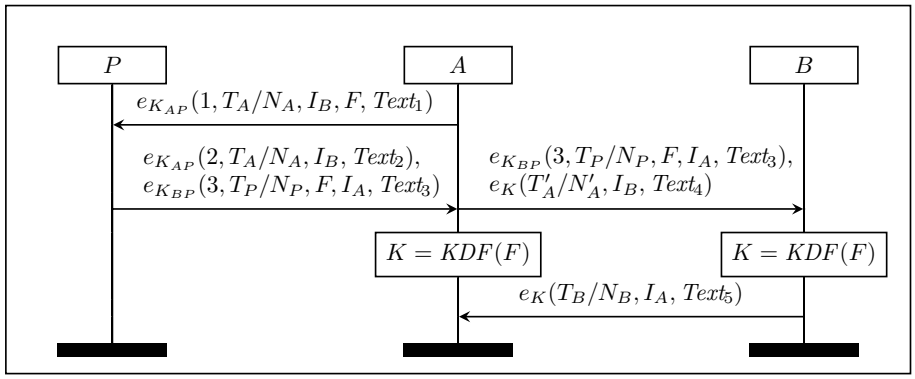
In this work, we focus on Part 2 [19] and Part 3 [20] of the ISO/IEC 11770 standard. Part 2 describes 13 key establishment mechanisms. Part 3 describes 11 key agreement mechanisms, 6 key transport mechanisms, and 3 public key transport mechanisms. Many of these 33 mechanisms have optional message fields and message flows, giving rise to a large number of variants.

Additionally, the mechanisms produce keying material that must be used with a key derivation function to form an encryption key for further messages. The standard does not specify a single key derivation function; instead, it gives examples of various possible key derivation functions. Thus, using a single mechanism with different key derivation functions can be regarded as having multiple variants of the same base mechanism. As we will see in Section 4.4, the choice of a key derivation function can influence the security of a mechanism.

Naming conventions. We provide a unique name for each base mechanism in the considered parts of the standard. We refer to the thirteen key establishment mechanisms from Part 2 as protocol 2-1, 2-2, ..., 2-13. We refer to the key agreement mechanisms in Part 3 as 3-KA-1, ..., 3-KA-11, to the key transport mechanisms as 3-KT-1, ..., 3-KT-6, and to the public key transport mechanisms as 3-PKT-1, 3-PKT-2, and 3-PKT-3.

We next describe two protocols from the standard. This enables us to introduce notation and provide an indication of the protocols contained in the standard.

Key Establishment Mechanism 12 (2-12). We give an example of a protocol described in Part 2 [19], referenced in the standard in Section 7.2 as Key Establishment Mechanism 12. The protocol is stated to be derived from, but not fully compatible with, the four-pass mutual authentication mechanism specified in ISO/IEC 9798-2 [18]. The protocol has several variants. For this example, we consider the variant with all optional parts included, depicted using a Message Sequence Chart (MSC) in Figure 1. In the figure, T_A/N_A is either a time stamp T_A or sequence number N_A of entity A. I_A and I_B respectively identify entities A and B. $e_K(m)$ denotes the encryption of the message m with the key K. The protocol assumes that entities A and B



Notation	Meaning
T_A	time stamp of entity A
N_A	sequence number of A
I_A	identifier of entity A
$e_K(m)$	encryption of m with K
K_{AB}	key shared by A, B
$Text_i$	text field
F	keying material

Fig. 1 Protocol 2-12 with optional parts

respectively share long-term symmetric keys K_{AP} and K_{BP} with a trusted third party P. $Text_1$ through $Text_5$ are text fields whose contents are not specified by the standard. F denotes keying material.

The protocol proceeds as follows. When a party A wants to communicate with another party B, it contacts trusted third party P. A generates fresh keying material F and includes it in the message encrypted for P, who responds with two encrypted messages. They are respectively encrypted with K_{AP} and K_{BP}. Both encrypted messages are sent to A, who forwards the second encryption to B. B decrypts the message and obtains the keying material F. A and B now both use a key derivation function to compute the session key K from F. We are only considering the protocol variant with optional parts, so the protocol proceeds with two messages that allow both entities to confirm to the other entity that they have successfully computed the key.

For the key derivation function (KDF), we consider two extremes from the KDFs described in the standard: at the one end, some KDFs take as input only F, whereas others include additional parameters, such as the identities I_A and I_B.

Key Agreement Mechanism 11 (3-KA-11). Key Agreement Mechanism 11 from Part 3, shown in Figure 2, establishes a key shared by entities A and B. First, A generates a random value r_A and sends it to B. B responds with his own random value r_B and his certificate. Upon receiving this message, A generates a new random value r'_A. r'_A is used with the other two random values to derive a session key K. Then r'_A is encrypted using B's public key, and sent to B along with a message authentication code (MAC) keyed with K that includes the earlier randomness r_A. B decrypts the message, computes K, and checks the MAC. B then responds with his own MAC of r_B and his certificate.

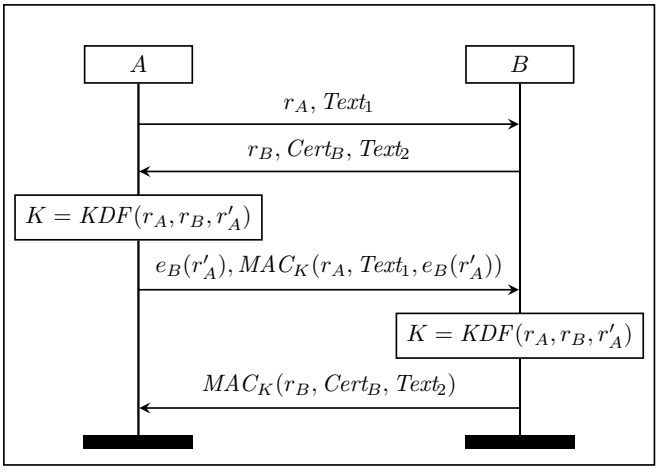


Fig. 2 Protocol 3-KA-11

According to the standard, this protocol is derived from the TLS Handshake Protocol [15]. In particular, since only B uses his private key (to decrypt the message) and the random values are directly input to the key derivation function, the protocol resembles TLS's unilaterally authenticated RSA mode, where A corresponds to the client and B to the server. The random value r'_A in 3-KA-11 plays the same role as TLS's *premaster secret* and the two text fields are used in TLS for the cipher suite negotiation.

2.2 Security properties and threat model of the standard

Most standards for security protocols do not specify threat models or intended security properties [5]. In this respect, ISO/IEC 11770 is an exception since it explicitly specifies a set of security properties, and states for each protocol which of these properties it satisfies. ISO/IEC 11770 defines the following properties [19, 20]:

Implicit key authentication from entity A to entity B Assurance for entity B that A is the only other entity that can possibly be in possession of the correct key.

Key confirmation from A to B Assurance for entity B that entity A is in possession of the correct key.

Explicit key authentication from entity A to entity B Assurance for entity B that A is the only other entity that is in possession of the correct key.[1]

Entity authentication of A to B Assurance of the identity of entity A for entity B.

Forward secrecy with respect to entity A The property that knowledge of entity A's long-term private key subsequent to a key agreement operation does not enable an opponent to recompute previously derived keys.

Forward secrecy with respect to A and B The property that knowledge of entity A's long-term private key or knowledge of entity B's long-term private key subsequent to a key agreement operation does not enable an opponent to recompute previously derived keys.

Mutual forward secrecy Property that knowledge of both entity A's and entity B's long-term private keys subsequent to a key agreement operation does not enable an opponent to recompute previously derived keys.

[1] The standard notes that "Implicit key authentication from A to B and key confirmation from A to B together imply explicit key authentication from A to B" [19, p. 2]. One might expect also that explicit key authentication implies the other two properties, but the standard does not state this.

For example, regarding the protocols described in the previous section, the standard claims the following: protocol 2-12 with optional parts satisfies mutual explicit key authentication, mutual key confirmation and mutual entity authentication, and protocol 3-KA-11 provides mutual explicit key authentication, mutual key confirmation, entity authentication to B and mutual forward secrecy.

The standard does not specify an explicit threat model. However, the security properties described above are not claimed for all protocols. Because some protocols apparently do not meet the above properties, we can conclude that the adversary is considered to have at least the following capabilities:

Injecting network messages Entity authentication is claimed for some, but not all mechanisms. Entity authentication can only be effectively violated if the adversary is able to inject or tamper with network messages.

Eavesdropping on network messages If the adversary could not eavesdrop on messages, we would need no complex key management mechanism, and could exploit simple authentication mechanisms.

Compromising long-term private keys Forward secrecy is claimed for some protocols. The adversary can only violate forward secrecy by compromising the long-term private keys of some entities.

3 Formally modelling the protocols and their properties

We analyse all 33 protocols specified in the standard, along with their described variants, by using formal methods. In particular, we use the Scyther framework [14] for the automatic symbolic analysis of security protocols.

The Scyther tool [12] has built-in support for compromising adversaries [2], including support for the analysis of (weak) Perfect Forward Secrecy, resilience against KCI attacks, and finding Unknown Key Share attacks. It is therefore especially suitable for analysing security notions common in the domain of protocols for key agreement, establishment, and transport. Another feature of Scyther which is very helpful in our research is its option-packed back end, which allows for a conveniently scriptable analysis of whole classes of security protocols.

3.1 Protocol specification

Within the Scyther framework, protocols are specified using so-called role scripts. A protocol can have any finite number of roles, and is run by entities who execute those roles. Entities may execute each role multiple times, and every role can be executed by any entity. We call each such role instance a session. We assume that, prior to protocol execution, every entity has generated or securely received a long-term asymmetric key pair consisting of a public and a private key, it has authentic and secret copies of all its long-term symmetric keys shared with other entities, and authentic copies of the public keys of all other entities.

Roles are specified as sequences of send, receive and claim events. Events have term parameters, where terms are constructed from role names, function names, variables, fresh values, and constants. Receive events correspond to pattern matching on incoming messages, and may therefore contain variables to store incoming payloads, and fresh values generated in previous send steps. Send events can contain fresh values, and variables that occur in previous receive steps. These variables are initialised before the send events are executed. We specify intended security properties using claim events.

For example, in Figure 3, we give the input for the Scyther tool that encodes protocol 2-12 described in Section 2.1. Send, receive and claim events are respectively specified with `send`, `recv`, and `claim`. Freshly generated values are declared with `fresh`, variables with `var`, user-defined types with `usertype`, and hash functions with `hashfunction`. Every function, constant, fresh value, and variable can have a different type, such as `Nonce` or a user-defined type such as `Integer`, `KeyingMaterial`, or `String`—the types are used to restrict the pattern matching in the execution of a receive event. The keyword `macro` can be used to define shorthands.

3.2 Specifying security properties

We model the following properties from the standard: *key authentication*, *key confirmation*, *entity authentication*, and *forward secrecy*. Additionally, we model *key compromise impersonation (KCI)* and *unknown key share (UKS)* attacks.

Implicit key authentication. According to the standard, implicit key authentication requires that if an entity A uses a protocol to establish a key K with entity B, then only A and B can learn the key. We model this by analysing the secrecy of K whilst allowing the adversary to impersonate any entity except for A and B. The possibility of impersonation is modelled by allowing the adversary to learn the long-term private key of any entity except for A and B.

Key confirmation. This property corresponds to one of the authentication properties in Lowe's hierarchy [23]. In particular, key confirmation from A to B corresponds

```
option "--partner-definition=2";

usertype KeyingMaterial;
usertype String;
usertype Integer;

hashfunction KDF;
const N1,N2,N3: Integer;

macro key = KDF(F);
macro sid = (A,B,key);

protocol 2-12-withOptional(A,B,P)
{
  role A
  {
    fresh TNA,TNA2: Nonce;
    fresh F: KeyingMaterial;
    fresh Text1,Text4: String;
    var Text2,Text5: String;
    var T: Ticket;
    var TNB: Nonce;

    claim(A,SID,sid);
    claim(A,Running,B,key);
    send_1(A,P,{N1,TNA,B,F,Text1}k(A,P));
    recv_2(P,A,{N2,TNA,B,Text2}k(A,P),T);
    send_3(A,B,T,{TNA2,B,Text4}key);
    recv_4(B,A,{TNB,A,Text5}key);

    claim(A,SKR,key);
    claim(A,Commit,B,key);
    claim(A,Alive,B);
  }
  role B
  {
    var TNP,TNA2: Nonce;
    var F: KeyingMaterial;
    var Text3,Text4: String;
    fresh TNB: Nonce;
    fresh Text5: String;

    recv_3(A,B,{N3,TNP,F,A,Text3}k(B,P),
               {TNA2,B,Text4}key);
    claim(B,SID,sid);
    claim(B,Running,A,key);
    send_4(B,A,{TNB,A,Text5}key);

    claim(B,SKR,key);
    claim(B,Commit,A,key);
    claim(B,Alive,A);
  }
  role P
  {
    var TNA: Nonce;
    var F: KeyingMaterial;
    var Text1: String;
    fresh Text2: String;
    fresh TNP: Nonce;
    fresh Text3: String;

    claim(P,SID,P);
    recv_1(A,P,{N1,TNA,B,F,Text1}k(A,P));
    send_2(P,A,{N2,TNA,B,Text2}k(A,P),
               {N3,TNP,F,A,Text3}k(B,P));
  }
}
```

Fig. 3 Scyther input file for 2-12 with confirmation messages and claimed properties.

to non-injective data agreement on the key, which we model with two claims: a Running claim in the A role and a Commit claim in the B role. If the Commit claim is executed, we require that the corresponding Running claim is executed as well: it must have the entities in reverse order, and the same contents (the entities are said to *agree* on the contents). It is called *non-injective* data agreement because replays are not considered.

Explicit key authentication. In contrast to implicit key authentication, explicit key authentication additionally requires that entities in fact compute the key. We model this by the previously defined key confirmation.

Recall that the standard only states that implicit key authentication and key confirmation imply explicit key authentication [19, p. 2], but not necessarily the other way around. Thus, if the standard is taken literally, this suggests that there exist protocols that offer explicit key authentication but that do not satisfy implicit key authentication or key confirmation. We attempted to find a formal interpretation of these concepts that makes the suggestion true, but failed. Thus, we define explicit key authentication as the conjunction of key confirmation and implicit key authentication.

Entity authentication. Entity authentication from A to B corresponds to aliveness [23]: an Alive claim for A is placed in the specification of role B. Whenever the claim is executed, the entity assumed to be performing the A role is required to have executed some event.

Forward secrecy. There are several definitions of forward secrecy in the literature, and it is not clear from the standard which property is intended. The *mutual forward secrecy (MFS)* notion from the standard seems to be closest to two common formal definitions. *weak Perfect Forward Secrecy (wPFS)* [2,13,21] requires that the adversary does not actively tamper with the session that he attacks, e.g. by injecting messages. In contrast, *(strong) Perfect Forward Secrecy (PFS)* allows the adversary to actively interfere with the messages received by the session under attack. Scyther directly supports checking both properties through its support of the LKRaftercorrect and LKRafter rules [2]. Our analysis reveals that the majority of protocols for which MFS is claimed in fact only achieve wPFS, and we therefore interpret MFS as wPFS.

Key compromise impersonation (KCI). Another desirable property of key exchange protocols is the resilience to KCI attacks [6], in which the adversary exploits his knowledge of the long-term private key of Alice to impersonate any entity in later communication with Alice. This property is modelled in Scyther by a session key secrecy claim of an entity whose long-term private keys the adversary is allowed to reveal.

Unknown key share (UKS). Unknown key share attacks are attacks in which only Alice and Bob know the session key K; however, Alice and Bob disagree on who they share K with [7]. For example, Alice correctly thinks K is shared with Bob, but Bob might think that

K is shared with Charlie. Even though the adversary does not learn the key in such attacks, using the key is not sufficient to authenticate subsequent messages: if Alice sends a message encrypted with K or accompanied by a MAC keyed by K, Bob will assume that the message came from Charlie. Similarly, Bob will send messages intended for Charlie that will be received by Alice.

We model UKS attacks in the standard way, i.e. if the assumptions on the partner identities of the attacked session s do not match the assumptions of a session s', we allow the adversary to reveal the session key of s'. This causes UKS attacks to manifest as violations of secrecy of the session key computed by s. Note that false positives can also occur, where the revealed session key is used for more than computing the session key of s, e.g. for injecting messages.

We enable session identifier (SID) support in Scyther input files with `option "--partner-definition=2"`, and we specify SIDs for all role instances by annotating each role with `SID` claims. For example, in Figure 3 we enable the manual specification of a partner session on line 1, define the session identifier on line 11 and insert it into role specifications on lines 24 and 45. When session key secrecy is analysed for a session s, and Scyther's SKR adversary rule (Session-Key Reveal) is enabled in the GUI or `--SKR=1` is provided as a command-line option, the adversary is able to obtain the session keys computed by any session whose identifier differs from that of s.

4 Results of the formal analysis

For the protocols in Part 2 and Part 3 of the standard, we model the claimed security properties as well as some additional properties that serve to sanity-check the models. Also, we consider KCI and UKS attacks.

4.1 Main analysis results

We analyse each of the resulting models in Scyther's default setting. This standard setting covers a wide range of scenarios, some of which may not apply to all real-world implementations. We will return to this below. The results are displayed in Tables 1–4. In the tables, we use a cross (✕) to denote that an attack was found, and a check (✓) to denote that no attacks were found. These tables were automatically generated using a script that uses the Scyther tool as a back end.

In the case that we find attacks for a specific protocol and property, we use further automated analysis to narrow down the scenarios in which the protocol is vulnerable. In particular, we consider the following aspects of an attack:

Agents in multiple roles. Some attacks require that an agent performs multiple roles. This behaviour is allowed by our formal models and in many application scenarios. However, there are application scenarios in which each agent performs only one role, and where such an attack would not apply.

Alice-talks-to-Alice. Some attacks require that an agent starts a role with itself in one of the other roles. This occurs, for example, in many implementations of the Kerberos protocol. However, such attacks would not work on all implementations.

In the tables, crosses imply that we find attacks that might rely on these non-standard, subtle requirements on attack scenarios. On the other hand, if no attacks are found, we consider the following strengthening of our threat model and include its impact on the standard in Table 5:

Type flaw. We say an attack requires a type flaw if it depends on an agent misinterpreting a term as a term of another type. For example, an agent may misinterpret an agent name as a nonce.

Note that not all crosses in the tables imply a serious flaw in the protocol. Rather, they indicate that a different protocol could have achieved these properties, perhaps at a reduced efficiency. Also note that we model some properties beyond those claimed in the standard. We analyse the exact discrepancies between our results and the claims in the standard in Section 4.2, where we also return to the implementation scenario assumptions required for the attacks.

Finally, we manually inspect the attack graphs generated by Scyther to sanity-check the results and to understand which aspects of a protocol's design make it vulnerable.

4.2 Implications for properties claimed in the standard

We give an overview of the properties claimed in the standard in Table 5. The contents of this table are directly taken from the tables in [19,20], with the difference that we add notes and use red and bold to mark incorrect statements, based on our formal analysis results. We classify the incorrect claims in the standard into five categories AT1...AT5, which we describe below.

Note that the table in [19] only has a key authentication column with "yes" or "no" in the cells, but this information has to be combined with NOTE 2 [19], which states that all protocols in Part 2 achieve implicit

Protocol	Optional fields	entity authentication		implicit key authentication	KCI resilience	key confirmation/ explicit key authentication	
		of A	of B			of A	of B
2-1		×		✓	×	×	
2-2		×		✓	×	×	
2-3		✓		✓	×	✓	
2-3	all	✓		✓	×	✓	
2-4		✓		✓	×	✓	
2-4	all	✓		✓	×	✓	
2-5	all	✓	✓	✓	×	×	×
2-6	Fa,Fb	×	×	✓	×	×	×
2-6	Fb	×	×	✓	×	×	×
2-6	B,Fb	✓	✓	✓	×	✓	✓
2-6	B,Fa	✓	✓	✓	×	✓	×
2-6	all	✓	✓	✓	×	✓	✓
2-7		×	×	✓	×	×	×
2-8	N2	×	×	✓	×	×	×
2-8	N2,MACN3	✓	✓	✓	×	×	×
2-9		×	×	✓	×	×	×
2-9	all	×	×	✓	×	×	×
2-10		×	×	✓	×		
2-11		×	×	✓	×		
2-12		×		✓	×	×	
2-12	all	×	✓	✓	×	×	×
2-13		×		✓	×	×	
2-13	all	×	×	✓	×	×	×

Table 1 Main formal analysis results for key establishment protocols from Part 2

Protocol	Optional fields	entity authentication		implicit key authentication		KCI resilience		key confirmation/ explicit key authentication		PFS		weak PFS	
		of A	of B	of A	of B	for A	for B	of A	of B	for A	for B	for A	for B
3-KA-1				✓	✓	×	×			×	×	×	×
3-KA-2				×	✓	✓	×			×	×	×	×
3-KA-3	TVP	✓		✓	✓	✓	×	✓		✓	×	✓	✓
3-KA-3		✓		✓	✓	✓	×	✓		✓	×	✓	✓
3-KA-4			×	×	×	×	×	×	×	×	×	×	×
3-KA-5	MAC		✓	✓	✓	✓	✓	×	✓	✓	×	✓	✓
3-KA-6	MAC	×	✓	✓	✓	×	✓	×	✓	×	×	×	✓
3-KA-6		×	✓	✓	✓	×	✓	×	✓	×	×	×	✓
3-KA-7		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
3-KA-8				✓	✓	✓	×			✓	×	✓	×
3-KA-9		×	×	✓	✓	✓	✓	×	×	×	×	✓	✓
3-KA-10		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
3-KA-11		×	✓	×	✓	✓	×	×	×	×	×	×	×

Table 2 Main formal analysis results for key agreement protocols from Part 3

key authentication, and that "yes" is to be interpreted as explicit key confirmation.

The standard provides a reasonable level of detail in its specification of security properties and assumptions, but does not provide sufficient detail to unambiguously construct a formal model. We have therefore chosen to focus on positively claimed properties and use the formal analysis to construct counterexamples in the form of attacks.

The benefit of this approach is that we can often exhibit straightforward attacks without having to argue about the full details of the assumed threat models, protocol execution model, and modelling properties. However, a drawback is that we cannot provide conclusive statements about other oddities in the standard. For example, according to the standard, protocol 2-5 has a "no" for key confirmation, but it has a "yes" for explicit key authentication. This seems to contradict the informal definitions of these notions in the standard:

Protocol	Optional fields	entity authentication		implicit key authentication		KCI resilience		key confirmation/ explicit key authentication		PFS	weak PFS	
		of A	of B	of A	of B	for A	for B	of A	of B		for A	for B
3-KT-1		×	×	×	✓	✓	×	×		×	×	×
3-KT-1	all	×	×	×	✓	✓	×	×		×	×	×
3-KT-2		✓	×	✓	✓	✓	×	×		×	×	×
3-KT-2	all	✓	×	✓	✓	✓	×	✓		×	×	×
3-KT-3		✓	×	✓	✓	✓	×	✓		×	×	×
3-KT-3	all	✓	×	✓	✓	✓	×	✓		×	×	×
3-KT-4	all	×	✓	✓	✓	×	✓	×	✓	×	×	✓
3-KT-5	MACN3	✓	✓	✓	✓	×	×	✓	✓	×	✓	✓
3-KT-6	noText	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×
3-KT-6		✓	✓	✓	✓	✓	✓	✓	✓	×	✓	×

Table 3 Main formal analysis results for key transport protocols from Part 3

Protocol	Optional fields	entity authentication		implicit key authentication		KCI resilience		key confirmation/ explicit key authentication		PFS		weak PFS	
		of A	of B	of A	of B	for A	for B	of A	of B	for A	for B	for A	for B
3-PKT-1		✓						✓					
3-PKT-2	N3 sign.	✓						✓					
3-PKT-2		✓						✓					
3-PKT-3								✓					

Table 4 Main formal analysis results for public key transport protocols from Part 3

from their definitions, one would expect protocols with explicit key authentication to also satisfy key confirmation. Future versions of the standard would benefit from a clarification of the exact relations between these properties.

AT1**: Entity authentication failures for 2-8, 2-9, 2-12, and 2-13.** We find several possible entity authentication failures for protocols in Part 2 that are derived from protocols in an earlier version of the ISO/IEC 9798-2 standard for entity authentication [18].

These attacks are closely related to the attacks on the corresponding protocols from the 9798 standard as presented in [4]. The attacks work in all implementations where a single entity can perform not only the role of the trusted third party, but also another role. The adversary can then cause A to complete the protocol, apparently with B, even though B is not present. Thus, the attacks violate even the weakest form of entity authentication.

We show an example of such an attack on protocol 2-12 in Figure 4. It depends on the fact that the entity running role A does not check the contents of the message encrypted for entities running roles B and P. In fact, normally such a check is impossible because all three roles are run by different entities. Seeing the payload of that particular message would be the only way for Pete to detect that something is wrong: he could see that the message contains I_{Alice} where I_{Pete} should be. Since he cannot see the payload, he gladly confirms the session key to Bob in role B, who falsely thinks that Alice just confirmed it.

Fixes for these protocols have been proposed in [4], and they have been integrated into the ISO/IEC 9798 standard. As a result, these attacks no longer work on ISO/IEC 9798. However, no changes have been made to the derived protocols in ISO/IEC 11770, so they are still vulnerable to similar attacks.

AT2**: 3-KA-11 key authentication/confirmation failure for B.** According to the standard, this mechanism (depicted in Figure 2) offers mutual explicit key authentication and mutual key confirmation. However, as stated earlier, 3-KA-11 is derived from the unilaterally authenticated RSA mode of TLS [15]. In this mode, the server cannot be certain whether the client is who he claims to be. The same issue occurs for the B role of 3-KA-11.

Consequently, there is an attack on entity authentication on the B role that violates both of the claimed properties. In the attack, the adversary performs the A role, pretending to be Alice, and sends messages to Bob in the B role. Because executing the A role does not require the use of any long-term secrets, the adversary can simply claim to be anybody. The entity performing the B role therefore cannot obtain any authentication guarantees about its communication partner or ascertain the secrecy of the key.

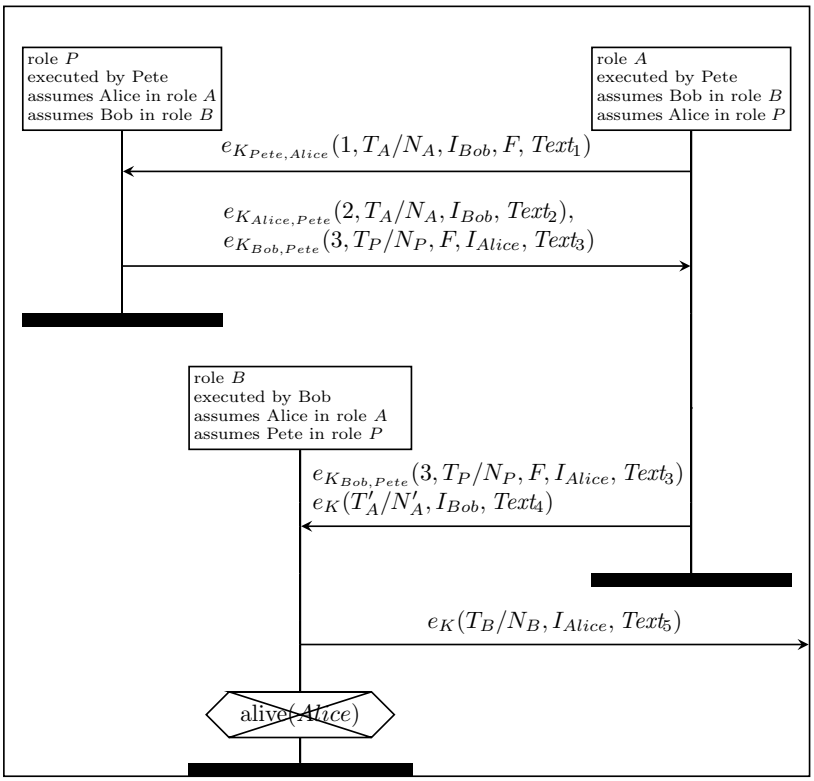


Fig. 4 Entity authentication attack on protocol 2-12 with optional parts

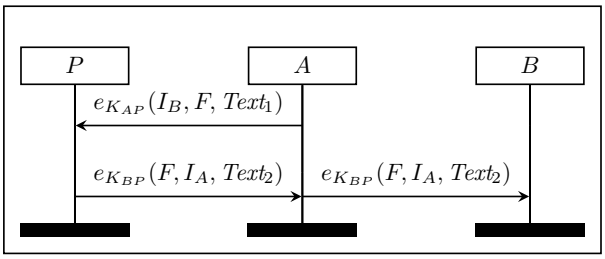


Fig. 5 Protocol 2-11

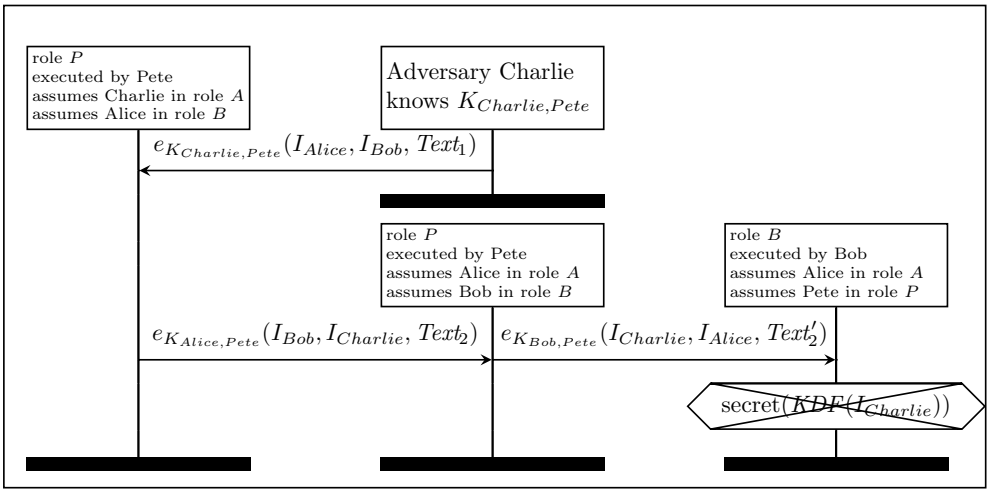


Fig. 6 Protocol 2-11 key authentication attack

Protocol in Part 2	Key Auth.	Key Conf.	Entity Auth.
2-1	implicit	no	no
2-2	implicit	no	no
2-3	explicit	no	A
2-4	explicit	no	A
2-5	explicit	no	A & B
2-6	explicit	no	A & B
2-7	implicit	no	no
2-8	**explicit** (AT1)	**opt.** (AT1)	**opt.** (AT1)
2-9	**explicit** (AT1)	**opt.** (AT1)	**opt.** (AT1)
2-10	explicit	no	no
2-11	**explicit** (AT4)	no	no
2-12	**explicit** (AT1)	**opt.** (AT1)	**opt.** (AT1)
2-13	**explicit** (AT1)	**opt.** (AT1)	**opt.** (AT1)

Protocol in Part 3	Implicit Key Auth.	Key Conf.	Entity Auth.	Forward Secrecy
3-KA-1	A,B	no	no	no
3-KA-2	B	no	no	A
3-KA-3	A,B	B	A	A
3-KA-4	no	no	no	MFS
3-KA-5	A,B	opt	no	A,B
3-KA-6	A,B	opt	B	B
3-KA-7	A,B	A,B	A,B	MFS
3-KA-8	A,B	no	no	A
3-KA-9	A,B	no	no	MFS
3-KA-10	A,B	A,B	A,B	MFS
3-KA-11	A,**B** (AT2)	A,**B** (AT2)	B	**MFS** (AT3)
3-KT-1	B	no	no	A
3-KT-2	B	B	A	A
3-KT-3	B	B	A	A
3-KT-4	A	A	B	B
3-KT-5	A,B	(A),B	A,B	no
3-KT-6	A,B	**A,B** (AT5)	A,B	no

Table 5 Claimed properties Security properties claimed for the protocols in Parts 2 and 3 of the standard. Our analysis reveals that some claims are incorrect, and we mark them using bold and red.

AT3: **Failure of MFS for 3-KA-11.** Because protocol 3-KA-11 is derived from the RSA mode of TLS, it provides no forward secrecy with respect to B. The adversary only needs to observe a regular session. If he afterwards obtains the long-term private key of B, he can decrypt $e_B(r'_A)$ and learn r'_A. Since r_A and r_B have been sent in plaintext, the adversary then has all the ingredients he needs to recompute the key K.

AT4: **Failure of key authentication for 2-11.**

The 2-11 protocol, which is depicted in Figure 5, assumes pre-shared symmetric keys and a trusted third party P. In a regular execution of the protocol, A sends a request to P for a ticket to forward to B. The request is a triple $(I_B, F, \mathit{Text}_1)$, which contains keying material F generated by A and is encrypted with the key shared between A and P. P then returns a triple $(F, I_A, \mathit{Text}_2)$ encrypted with the key shared between B and P, which A forwards to B.

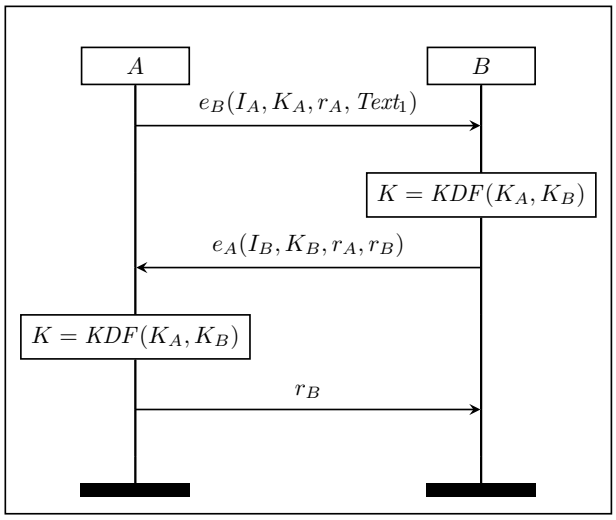


Fig. 7 Protocol 3-KT-6 combined key variant with Text_1 optional field

Depending on the implementation, it may be possible for an agent to misinterpret an agent identity as (random) keying material, for example if both are of the same bit length. If an implementation of 2-11 cannot tell the difference between these, it can be vulnerable to a type-flaw attack on key authentication. The attack can be seen in Figure 6.

The adversary Charlie can attack a session in which Bob assumes to be talking to Alice, even though Alice and Bob are not compromised. Charlie encrypts a message for the trusted third party Pete, requesting a key for Alice. However, instead of generating new keying material F, Charlie instead includes Bob's identity in the keying material field. Pete's response therefore is the triple $(I_{\text{Bob}}, I_{\text{Charlie}}, \mathit{Text}_2)$ encrypted with $K_{\text{Alice,Pete}}$. Charlie resends this message to Pete. There is nothing in the standard that prevents Pete from accepting this message as a valid request. Now, Pete responds with the triple $(I_{\text{Charlie}}, I_{\text{Alice}}, \mathit{Text}'_2)$ encrypted with $K_{\text{Bob,Pete}}$. If Bob receives this message, he will assume that it is a valid message and that I_{Charlie} is secure keying material for communicating with Alice. The adversary can then compute the session key that Bob computes.

AT5: **Failure of key confirmation for 3-KT-6.** The 3-KT-6 protocol is a three-pass protocol that transfers two secret keys, K_A and K_B. After the exchange, a session key can be computed from either or both of these keys. There are five text fields designated as optional in the protocol's specification. We choose to depict a simple implementation with only Text_1 enabled in Figure 7.

A complex attack is possible on some implementations of this protocol. There are three preconditions for

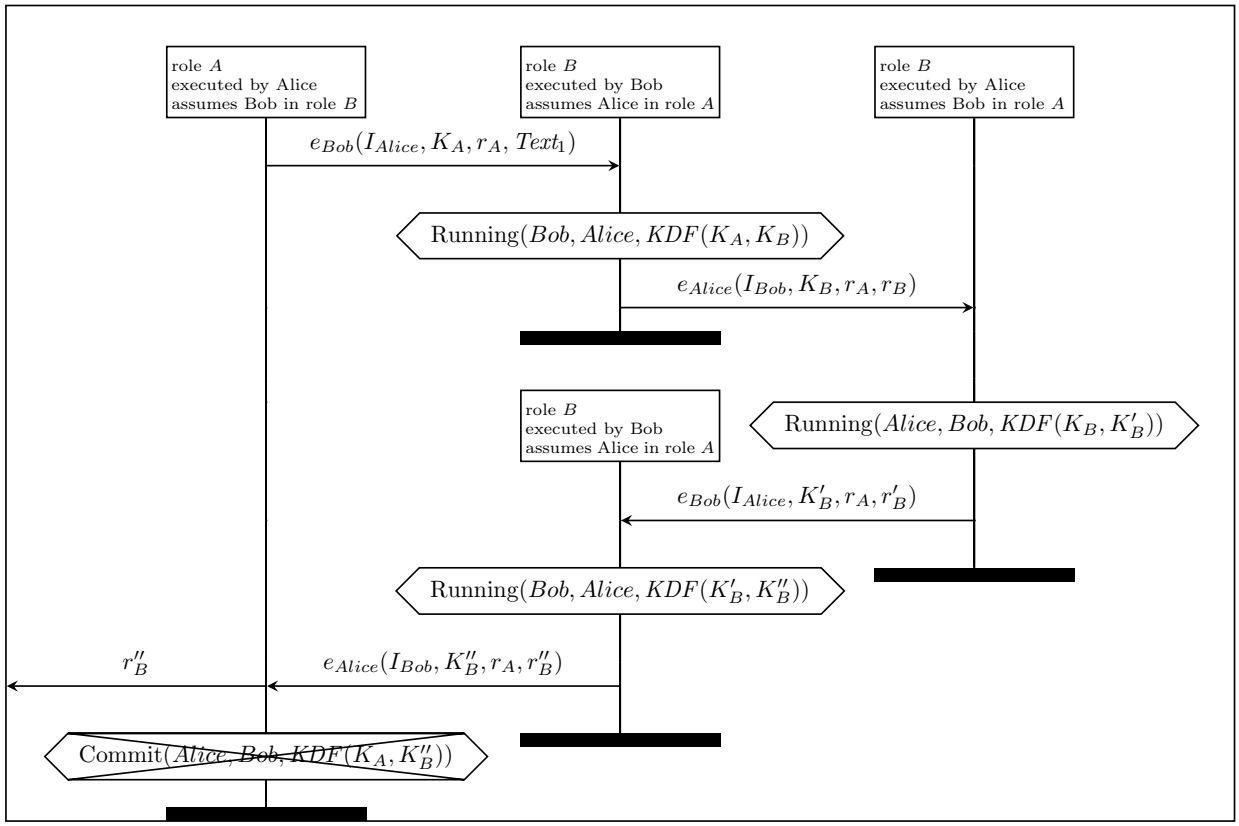


Fig. 8 3-KT-6 combined key variant with $Text_1$ optional field key confirmation attack

the attack, which will not be met by most implementations. However, there is nothing in the standard that ensures that they are not met. The first precondition is that the $Text_1$ field is implemented, and $Text_3$ is not implemented. Second, fresh values must be acceptable values for the $Text_1$ field. Third, entities must be able to perform both the A and B role of the protocol.

If an implementation meets these conditions, the adversary can attack an instance of the A role by exploiting three instances of the B role. We give a graphical representation of such an attack in Figure 8. The adversary redirects each sent message into the first receive of a new instance of the B role, and the entity assumptions for the next instance of B are swapped. This is possible since entities can perform multiple roles, and enabled by the fact that the fresh values in messages sent by instances of the B role can be accepted into the $Text_1$ field. After three instances of the B role, the final message is then rerouted back to the final receive of the A instance. Consequently, there is no instance of B that agrees with the A instance on *both* K_A and K''_B. Thus, when the session key is computed from both of these keys, key confirmation fails for the instance of A.

4.3 Key Compromise Impersonation (KCI) results

All of the protocols in Part 2 use symmetric cryptography and hashing only. Hence, they are necessarily vulnerable to KCI attacks, which is implied by the impossibility result from [3]. All the modelled security properties of key transport protocols from Part 3 that are not satisfied can be violated even without allowing KCI. In some sense, we can consider all these attacks to be false positives of KCI attacks [3], which is the view we adopt in the continuation of our KCI discussion.

The automatic analysis shows that four of the eleven key agreement protocols in Part 3 are vulnerable to KCI attacks: 3-KA-1, 3-KA-3, 3-KA-6, and 3-KA-8. Mechanisms 3-KA-1 and 3-KA-3 are variants of the unsigned Diffie-Hellman protocol. Mechanism 3-KA-1 is the static Diffie-Hellman protocol, so as expected the session key is not secret when the adversary knows one of the static keys. Similarly, 3-KA-3 is a one-pass Diffie-Hellman variant where A's ephemeral and B's static half keys are used: if the adversary gets B's static private key, he can use A's half key to infer the session key.

In 3-KA-6, the fact that the input to the key derivation function is only protected by the public key of A allows an adversary who knows A's private key to impersonate B in subsequent communications with A

that are only protected by the established session key. Lastly, 3-KA-8 is derived from one-pass MQV [22]. The adversary can use B's private key to infer the session key computed by B without tampering with the message which B gets from A.

None of the KCI attacks that we find in this set of protocols require the adversary to use the actor's key to interfere before the attacked session ends. As a result, the attacker can delay the use of the actor's key until after the attacked session ends, and then use it to compute the session key. This means that the KCI attacks we find can also be regarded as attacks on wPFS (and thus PFS). In other words, all protocols in the standard that satisfy wPFS or PFS, also satisfy KCI resilience.

This observation about the standard may lead to the hypothesis that *all* protocols that satisfy PFS are also KCI resilient. However, this is not the case. For example, consider the 3-message version of the Unified Model (UM) protocol as described in [26]. This protocol is standardised in the NIST standard SP 800-56A [1]. The protocol is based on Diffie-Hellman: the session key derivation includes the ephemeral DH key (g^{xy}) based on the exchanged DH values g^x and g^y. As shown in [26], this helps to ensure that the protocol satisfies PFS. However, it is not KCI resilient because the exchanged DH values are authenticated by a MAC whose key depends critically on the static DH key g^{ab}. If the adversary obtains the actor's long-term key, he can compute g^{ab} and authenticate DH values of his choosing. Inserting a message that contains a DH value g^z, where z is known to the adversary, leads to a KCI attack. Note that [26] explicitly excludes KCI resilience from its adversary model. Thus, this protocol proves that PFS does not imply KCI resilience in general.

However, for all the protocols in the ISO/IEC 11770 standard, PFS does imply KCI resilience. Hence, we can replace each protocol vulnerable to KCI attacks with one that achieves all the already satisfied security guarantees, plus forward secrecy with respect to both entities, by following the Forward Secrecy column in Table 5:

- 3-KA-1 can be replaced by 3-KA-5 (optionally, key confirmation can be enabled),
- 3-KA-3 and 3-KA-6 can be replaced by 3-KA-7 (if entity authentication is required) or 3-KA-5 (otherwise), and
- 3-KA-8 can be replaced by 3-KA-9.

4.4 Unknown Key Share (UKS) results

We use Scyther to analyse all protocols for which key authentication is claimed for UKS vulnerabilities. We find that UKS attacks are possible on every implementation of two such protocols, and on some implementations of several other protocols.

We first explain the unknown key share attack on the 3-KA-11 protocol in detail. A graphical representation is given in Figure 9. In the attack, the adversary does not modify the contents of any messages, but only changes the implicit sender/recipient fields. When Alice executes role A with her intended partner Bob, she sends out her first message. The adversary modifies the sender field to "Charlie" and forwards the message to Bob. Bob assumes Charlie wants to communicate with him, so Bob starts to execute the B role and sends the response message to Charlie. The adversary redirects this message to Alice. The protocol continues as usual, except that the adversary continues to modify the sender fields and redirecting the responses. There is nothing in the messages that allows the entities to check each other's beliefs about their communication partner. In the end, Alice and Bob compute the same key K. Although the adversary does not know this key, Bob will believe that any subsequent messages he receives, which are encrypted or authenticated using K, are coming from Charlie, when in fact they come from Alice. This can lead to a serious authentication flaw [8, p. 139].

Although 3-KA-11 is derived from the TLS protocol, the TLS protocol is not vulnerable to unknown key share attacks. The reason for this is that the TLS protocol performs confirmation on all previously received messages, which in TLS contain the identities of the sender and recipient. This confirmation will fail if the parties have different views on their communication partners. In some sense, 3-KA-11 can be regarded as a stripped down version of the unilateral TLS-RSA handshake where security-relevant information (the identities of the participants) has been removed.

A second UKS attack is possible on the 2-10 protocol, which suffers from a role-mixup attack where Alice and Bob both perform the A role and compute the same session key. This can lead to later reflection attacks and misinterpretation attacks when the session key is used to encrypt payloads. In implementations in which entities can perform multiple roles, protocols 2-2, 2-8, 2-9, 2-11, and 2-12 are also vulnerable to UKS attacks.

Fortunately, UKS attacks can be prevented by choosing a key derivation function that includes the identifiers (I_A and I_B) of the involved entities [7, 8]. For example, this is required by the NIST SP-800-56A key derivation [1], which is included in Part 3 of the standard.

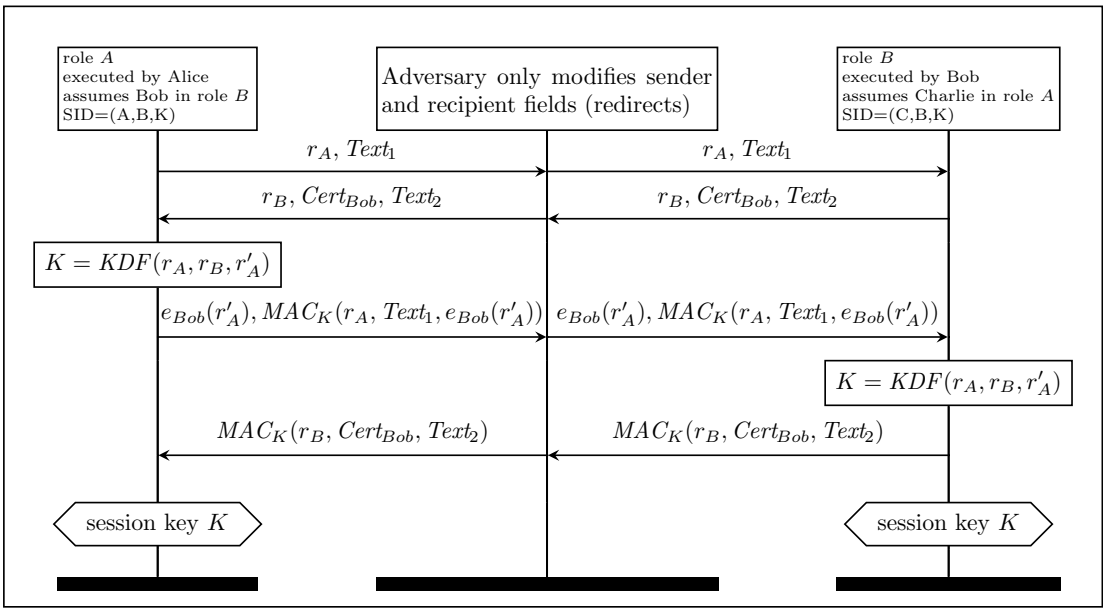


Fig. 9 Unknown key share attack on protocol 3-KA-11. Alice shares a key K with Bob as she expects, but Bob mistakenly assumes he shares K with Charlie.

We modelled the use of this KDF and used automated analysis to confirm that this prevents the UKS attacks. Intuitively, including the identities in the KDF ensures that entities that have different beliefs about their intended peers compute different keys, which thwarts UKS attacks.

5 Recommendations

In this section, we provide four recommendations to improve the ISO/IEC 11770 standard.

1. Making the threat model explicit. It is commendable that for every protocol in this standard there is a list of fairly precisely defined security requirements. However, an essential, yet missing ingredient to unambiguously state what is meant by these properties is an explicit threat model. Without the threat model, it is impossible to assess if the security requirements are met, as also discussed in [5]. We recommend its addition to the standard, with the proviso that any introduced differences from our threat model might require additions to or revisions of our other recommendations.

2. Improving protocols to achieve stated properties. Our second recommendation is to make small changes to the protocols to achieve their stated properties, if possible. The most straightforward way is to adopt the recommendations made for ISO/IEC 9798 in [4, p. 14]. In particular, we require that

- no cryptographic data should be interchangeable, which can be enforced by including unique tags,
- when optional fields are not used, then they must be set to empty, and
- entities that perform the role of the TTP in the 2-8, 2-9, 2-12, and 2-13 protocols must not perform the A or B role.

Following these recommendations addresses all of the issues in Table 5 except for the problems with protocol 3-KA-11.

3. Using appropriate key derivation functions. Our third recommendation improves the security of the standard by preventing unknown key share attacks. If the input to the key derivation function includes the identities of the communicating parties, UKS is directly prevented. For example, the execution of protocol 3-KA-11 depicted in Figure 9 no longer constitutes a UKS attack: Alice and Bob simply compute different session keys. We therefore recommend making the inclusion of identities in the key derivation an explicit requirement. A key derivation function from NIST SP-800-56A [1], which is described in ISO/IEC 11770, meets this requirement.

4. Addressing remaining issues with 3-KA-11. 3-KA-11 inherently does not offer perfect forward secrecy or mutual authentication. Switching to a protocol that does, such as mutually authenticated TLS-DHE_RSA, substantially changes the environmental assumptions, including the pre-distribution of keys.

A simpler solution is to adapt the statements made about the protocol. In particular, it should *not* be claimed in the overview table [20, p. 42] that 3-KA-

11 achieves implicit key authentication for both entities, that it achieves key confirmation for both entities, or that it achieves MFS. Similarly, the running text [20, p. 26] should not claim that 3-KA-11 achieves mutual explicit key authentication.

6 Related work

In 1998, Horng and Hsu presented an attack on an early version of the 3-KT-6 protocol [17]. This version contained no identity I_B of B in the second message, which enabled an attack similar to the 1995 attack by Gavin Lowe on the Needham-Schroeder protocol [24]. The attack on this version of 3-KT-6 violated key confirmation and showed that the protocol did not offer any strong mutual authentication. In the same year, Mitchell and Yeun proposed a fix [27] that was later introduced in the standard. They essentially performed Lowe's Needham-Schroeder fix by adding I_B to the second message.

In 2004, Cheng and Comley presented two attacks on a previous version of the 2-12 protocol [10,11]. Their first attack is a replay attack that depends on compromising session keys of threads not under attack, and the fact that random or sequence numbers are used where timestamps would be appropriate. Cheng and Comley fixed the protocol by replacing the used sequence numbers with timestamps.

A second attack is possible even when timestamps are used. It is a type-flaw attack based on the possibility of interpreting an identity field as a fresh key. The protocol was fixed by cryptographically binding the two parts of the second message (the second part became part of the payload encrypted to form the first part).

Initially, protocol 2-12 was withdrawn from the standard, but it was later updated in 2008 with a new version that did not suffer from these attacks. This new version is replay-protected by tagging with constants [19, p. 17], so that a mixup of messages can no longer occur. Since the type-flaw attack was also essentially a replay attack, it was automatically prevented as well.

Mathuria and Sriram used Scyther to discover in 2008 [25] more complex type-flaw attacks on a modified version of protocol 2-13 and on Cheng's and Comley's proposed fixed protocol. The attacks relied on the possibility that complex fields (concatenations, encryptions) could be interpreted as atomic fields (random values, keys, identities) in some implementations. While the first attack did apply to the 2-13 protocol itself, the second one did not apply to the updated version of 2-12, because this version of 2-12 was not based on fixes from [10, 11].

In 2010, Chen and Mitchell [9] generalised some of the concepts occurring in this class of type-flaw attacks and presented countermeasures, some of which found their way into later versions of ISO standards. They called the generalised phenomenon *parsing ambiguity attacks* and showed how many of these attacks can be found in the then current versions of ISO/IEC 11770 and ISO/IEC 9798. We continued their work by systematically analysing all the protocols in Parts 2 and 3 of the current version of the ISO/IEC 11770 standard.

7 Conclusions

Commendably, the ISO/IEC 11770 standard includes statements about the security guarantees achieved by its protocols, such as those reflected in Table 5. It is currently rare for a standard to include such statements. Specifying such security guarantees substantially helps the users of the standard in selecting the appropriate protocol for a given scenario. We recommend that other standards follow this example and try to include more precise statements about the intended security guarantees in their specifications.

However, there exist attacks which render some of the statements in the standard false. In retrospect, though we found all the attacks through automatic analysis, some attacks should have been found by manual inspection. This holds especially for 3-KA-11, which is based on TLS's unilaterally authenticated RSA handshake: it is clear that this protocol cannot offer key authentication or confirmation for both parties, since only one party is authenticated.

One way in which standardisation bodies could be more proactive is by being aware of analyses of standards on which they build. For example, many protocols in ISO/IEC 11770 are mentioned to be derived from authentication protocols in ISO/IEC 9798. In 2012, the ISO/IEC 9798 standard was analysed, several problems were identified [4], and it was subsequently updated to fix the identified problems. However, it seems that no attempt was made to determine if the derived protocols inherited these problems. Our analysis shows that this was in fact the case, implying that the attacks on protocols from Part 2 of ISO/IEC 11770 could have been identified earlier. In fact, applying the recommendations for ISO/IEC 9798 as described in [4] to ISO/IEC 11770 would have prevented all of the issues in Table 5 except for those with 3-KA-11.

Standards that cover protocols for a wide range of different usage scenarios benefit from periodic updates with modern security requirements. The standard currently does not claim resilience to UKS or KCI attacks. One could consider identifying the protocols that achieve

these properties and improving the other protocols. For example, all UKS attacks that we found can easily be prevented at negligible cost by using key derivation functions that include the identities of the participants. We therefore recommend including the identities in the input to the KDFs in the standard.

Compared to other security protocol standards, ISO standards have been less analysed in the academic literature. A possible reason for this difference is that people who are not members of the working groups can only access the standards by purchasing the final versions. One possible way to promote the external analysis of ISO standards is to publish early drafts of proposed changes or new standards. Parties that are interested in applying the standards will still need to purchase the final versions to ensure they comply. However, interested parties can freely analyse the designs from the early drafts, which may help identify and prevent problems before the standards are deployed.

References

1. E. Barker, D. Johnson, and M. Smid. NIST SP 800-56A: Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised), 2007.
2. D. Basin and C. Cremers. Modeling and analyzing security in the presence of compromising adversaries. In *Computer Security - ESORICS 2010*, volume 6345 of *LNCS*, pages 340–356. Springer, 2010.
3. D. Basin, C. Cremers, and M. Horvat. Actor key compromise: Consequences and countermeasures. In *Proc. of the 27th IEEE Computer Security Foundations Symposium (CSF)*, pages 244–258. IEEE Computer Society, 2014.
4. D. Basin, C. Cremers, and S. Meier. Provably repairing the ISO/IEC 9798 standard for entity authentication. *Journal of Computer Security*, 21(6):817–846, 2013.
5. D. Basin, C. Cremers, K. Miyazaki, S. Radomirovic, and D. Watanabe. Improving the security of cryptographic protocol standards. *IEEE Security & Privacy*, 2014.
6. S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In *IMA Int. Conf.*, pages 30–45, 1997.
7. S. Blake-Wilson and A. Menezes. Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol, 1999.
8. C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography. Springer, 2003.
9. L. Chen and C. J. Mitchell. Parsing ambiguities in authentication and key establishment protocols. *Int. J. Electron. Secur. Digit. Forensic*, 3(1):82–94, 2010.
10. Z. Cheng and R. Comley. Attacks on an ISO/IEC 11770-2 key establishment protocol. Cryptology ePrint Archive, Report 2004/249, 2004. `http://eprint.iacr.org/`, retrieved on June 1, 2014.
11. Z. Cheng and R. Comley. Attacks on an ISO/IEC 11770-2 key establishment protocol. *I. J. Network Security*, 3(3):290–295, 2006.
12. C. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Proc. CAV*, volume 5123 of *LNCS*, pages 414–418. Springer, 2008. Available for download at `http://www.cs.ox.ac.uk/people/cas.cremers/scyther/index.html`.
13. C. Cremers and M. Feltz. Beyond eCK: perfect forward secrecy under actor compromise and ephemeral-key reveal. *Designs, Codes and Cryptography*, pages 1–36, 2013.
14. C. Cremers and S. Mauw. *Operational Semantics and Verification of Security Protocols*. Information Security and Cryptography. Springer, 2012.
15. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) protocol version 1.2. IETF RFC 5246, August 2008.
16. European Payments Council. Guidelines on algorithms usage and key management. Technical report, 2009. EPC342-08 Version 1.1.
17. G. Horng and C.-K. Hsu. Weakness in the Helsinki protocol. *Electronics Letters*, 34:354–355(1), February 1998.
18. International Organization for Standardization, Genève, Switzerland. ISO/IEC 9798-2:2008, Information technology – Security techniques – Entity Authentication – Part 2: Mechanisms using symmetric encipherment algorithms, 2008. Third edition.
19. International Organization for Standardization, Genève, Switzerland. ISO/IEC 11770-2:2008, Information technology – Security techniques – Key Management – Part 2: Mechanisms using Symmetric Techniques, 2009. Incorporating corrigendum September 2009.
20. International Organization for Standardization, Genève, Switzerland. ISO/IEC 11770-3:2008, Information technology – Security techniques – Key Management – Part 3: Mechanisms using Asymmetric Techniques, 2009. Incorporating corrigendum September 2009.
21. H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. Cryptology ePrint Archive, Report 2005/176, 2005. `http://eprint.iacr.org/`, retrieved on June 1, 2014.
22. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28:119–134, 2003.
23. G. Lowe. A hierarchy of authentication specifications. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW)*, pages 31–44. IEEE, 1997.
24. G. Lowe. An attack on the Needham-Schroeder public key authentication protocol. *Information Processing Letters*, 56(3):131–136, November 1995.
25. A. Mathuria and G. Sriram. New attacks on ISO key establishment protocols. *IACR Cryptology ePrint Archive*, 2008:336, 2008.
26. A. Menezes and B. Ustaoglu. Security arguments for the UM key agreement protocol in the NIST SP 800-56a standard. In M. Abe and V. D. Gligor, editors, *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, Tokyo, Japan, March 18-20, 2008*, pages 261–270. ACM, 2008.
27. C. J. Mitchell and C. Y. Yeun. Fixing a problem in the Helsinki protocol. *SIGOPS Oper. Syst. Rev.*, 32(4):21–24, Oct. 1998.
28. L. Schmid. Improving the ISO/IEC 11770 standard, 2013. Bachelor's thesis, ETH Zurich, Switzerland.