Improving the ISO/IEC 11770 standard

Lara Schmid, Supervisor: Cas Cremers

September 4, 2013

Abstract

We analyze Part 2 and Part 3 of the ISO/IEC 11770 standard for key management and explain how the protocols can be modeled in the language of the automated security protocol analysis tool Scyther. Then, we provide an evaluation of the results, revealing attacks already discovered in previous work, as well as new ones. Also, we analyze the secrecy of the protocols with respect to different adversary-compromise rules to present a hierarchy of the results. In the end, we suggest changes to clarify the specifications in the standard, as well as fixes and improvements to the protocols to make them more secure.

Contents

1 Introduction

In information technology, cryptographic functions are needed to provide authentication of the communicating parties, as well as to protect the confidentiality of the data transmitted. An important part in enabling such environments are the keys. The purpose of key management is to describe how entities can store, exchange and agree upon keying material for symmetric and asymmetric cryptographic mechanisms.
The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) provide together the standard 11770 which presents key management mechanisms. It is important to have detailed specifications of protocols which are not vulnerable to attacks, because the standard is internationally consulted when new protocols are designed.
In [12], several attacks and vulnerabilities have been found in the protocols of Part 2 and 3 of the standard ISO/IEC 11770 with the automated proof tool Tamarin. We want to extend these preliminary results to find out if maybe Scyther is a more suitable tool to analyze the protocols, since it has already been successfully used to improve for example the standard ISO/IEC 9798 in [4].
We give a detailed analysis with respect to a Dolev-Yao adversary and restrict the analysis with respect to other adversary models, because of time limitations. We assume that the reader has access to the standard ISO/IEC 11770, as this work is a companion to the standard.

Organization. In sections 2 and 3 we present the standard and how the protocols are modeled in the Scyther tool; in Section 4 we introduce the models and what the standard claims about the protocols. The attacks found with respect to a Dolev-Yao adversary and other adversary models are discussed in the sections 5 and 6 respectively. In Section 7 and 8 we compare the results to the claims of the standard and provide recommendations for improving the standard. In Section 9 we compare our findings with previous work, in Section 10 we make suggestions for future work and in Section 11 we draw a final conclusion.

2 ISO/IEC 11770 Standard

2.1 Overview and naming scheme

In this section, we give a short overview of the ISO/IEC 11770 standard ([11], [9], [10]) for key management. The standard contains several parts. Part 1 is a framework and introduces definitions and concepts, Part 2 [9] introduces 13 key establishment mechanisms that use symmetric techniques, and Part 3 [10] presents 18 mechanisms using asymmetric techniques. This thesis covers the protocols of Parts 2 and 3.
In Part 2, the standard distinguishes between three kinds of mechanisms: protocols that use point-to-point key establishment (PtP), protocols that use a key distribution center (KDC) and protocols which use a key translation center (KTC). A KDC generates/acquires keys and distributes them, while a KTC enables keys to be transferred between pairs of entities that both share a key

1. B $\rightarrow$ A: R_B
2. A $\rightarrow$ B: $\{R_B \| I_B \| F \| Text_1\}_{K_{AB}}$

Figure 1: Protocol Isoiec-11770-2-4-a, with the optional I_B

with the KTC. In Part 3, the distinction is made between key agreement (KA) mechanisms, key transport (KT) mechanisms and public key transport (PKT) mechanisms.
As the naming in the standard is not consistent in the two parts, we introduce the following nomenclature: The first number after the standard name always denotes the part of the standard in which the protocol is presented. The protocols in the second part are enumerated, which is reflected in the second number (this can be seen in Table 1 on page 11). As in the third part for every group of protocols the counting starts freshly, there is a shortcut KA, KT or PKT and the enumeration within this section (Table 2 on page 12). In both cases, additional letters denote that there are variants of the protocol (see Section 2.3). For example Isoiec-11770-3-KT-2-a denotes the second protocol in the section "Key Transport" of the third part of the standard (with the variant a)).

2.2 Notation

Following the standard, we write $X \| Y$ to denote the concatenation of the bit strings X and Y. In contrast to the standard, we write $\{m\}_{K_{AB}}$ for the encryption of the message m with the symmetric key K_{AB}, $\{m\}_{pk(I)}$ for the encryption of m with the public key of an entity I and $\{m\}_{sk(I)}$ for the signature with a secret key of an entity I. In the symmetric protocols, bidirectional keys K_{AB} are used; there is no difference between K_{AB} and K_{BA}. Further, the conventions from the standard are followed that MAC denotes a Message Authentication Code and TVP_X denotes a Time Variant Parameter (issued by entity X) which is a data item such as a random number, a sequence number or a time stamp. T_X/N_X and R_X denote a time stamp or a sequence number and a random number (issued by entity X) respectively.

2.3 Variants of the Protocols and their Naming

There are several protocols with different variations. Often, a distinguishing identifier from an entity can optionally be left away, resulting in a protocol which is secure under another adversary assumption. To illustrate this we show the protocol Isoiec-11770-2-4 in Figure 1. The standard explains that the distinguishing identifier I_B in message 2 is optional, and if it is left away the protocol is only secure in an environment where substitution attacks are not possible. Where there are variants, we use "a" to refer to the base version and other characters for the variants. This letter is added to the name, resulting in Isoiec-11770-2-4-a and Isoiec-11770-2-4-b in this example. Where there are several omissions possible, in variant "b" all of them were left away, but all of the combinations have been analyzed and commented if they make a difference. Sometimes, an additional variant is described, where more letters are necessary to make the distinction. For all of these cases a detailed description of the variants can be found in Section 4.1.
A special case of data items that give rise to variations are the text fields. In

the second part of the standard [9], it is mentioned in Section 4 that *"The fields Text1, Text2, ..., specified in the mechanisms can contain optional data for use in applications outside the scope of this part of ISO/IEC 11770 (they can be empty). Their relationship and contents depend upon the specific application. One such application is message authentication."* The text fields are modeled as the most general data type, and the variations of omitting them or using them for message authentication are only modeled when this option is explicitly mentioned in the protocol description. In all the protocols of Part 2, the description never mentions the text fields, but there are always included in the sketch of the protocol and hence in the model. Unlike in Part 2, in Part 3 of the standard, text fields also appear outside encrypted messages and give rise to attacks in some cases. The variations where they are omitted are, therefore, modeled when this option is explicitly mentioned.
Another naming variation arises from the choice of key derivation function, which is explained in Section 3.2.
When clear from the context, we omit the "Isoiec-11770-" prefix for brevity.

2.4 Threat Model in the Standard

The standard does not specify a threat model. For some of the protocols, security property statements such as *"unilateral authentication from A to B given"* are made. It is not specified, however, under which assumptions these properties hold or what the exact interpretation of *"unilateral authentication"* should be. For many protocols no such statements are made.

2.5 Attack types considered in Standard

The attack types described in the standard are the following:

Substitution Attacks: A substitution attack is, according to the standard [9], when an adversary reuses legitimate messages sent by the initiator or the responder to masquerade as one of them.

Reflection Attacks: Reflection attacks are in [9] explained to be a specific form of substitution attack where a message sent by an entity is sent back to him by a masquerading adversary, in order to convince the sender that he is talking to the intended responder.

Replay Attacks: If an adversary replays a message in a later point of time, this constitutes a replay attack.

3 Modeling the Protocols for Scyther

3.1 Scyther tool

Scyther is an automated security protocol analysis tool which can be downloaded from [5], also see [7]. We refer to [3] and [4] for previous papers where Scyther has been used for analysis. Since the modeling can best be explained on example, we show in Figure 7 on page 46 the input model of the protocol 2-4-a, which is presented in Figure 1. As can be seen in Figure 7, the protocols

are modeled from the perspective of the participating entities, A and B in this example. For each of them, a new role is introduced which first contains a declaration of values. These can be fresh, that is this role generated them, or they can be of type variable, if the value is sent by another entity. Each of the declarations also includes a type, which is one of the following: *SessionKey*, *Nonce* for modeling time variant parameters or *Ticket* which is the most general form for modeling text fields. With the possibility of Scyther to create new, so called "user types", the type *Keying Material* was introduced, which is also used in the example model in Figure 7 on page 46.
The message exchange is again modeled in the different roles (see Figure 7): in the role of A it is declared how it interprets the first message received as a random number and what it answers in the second message.
Additionally, so called *claims* are made in each role. These specify the security properties that Scyther will evaluate for the protocol. The general form is *claim(I,Claim)* where *I* is the identifier of the role the claims is made in and *Claim* denotes the type of claim. Depending on the kind of claim, this is optionally followed by an argument, as can be seen in the claims in Figure 7. The claims used in this work can be found in Section 3.3.
By default, Scyther does an automated analysis of the models with respect to a Dolev-Yao adversary model (used for results presented in Section 5), but other adversary compromise rules can be chosen (discussed in Section 6). After running Scyther, a table is shown with an output for each claim made. Since it is in general an undecidable problem if there are any attacks, either of three outputs are possible: "verified", "fail" or "no attacks found within bounds". The case where "no attacks are found within bounds" denotes that Scyther stopped the search and could not find any attack so far. In the case where it is falsified, a button is displayed which shows a sketch of the attack.

3.2 Modeling Choices

For most of the data items the corresponding type in the Scyther model has been straight forward. However, there are concepts not specified or for which Scyther does not have a corresponding mechanism, which are treated in this section.

Bidirectional Symmetric keys: It is said in the standard [11] (p.5) that symmetric techniques *"use the same secret key for both the originator's and the recipient's transformation."* This corresponds to so called "bidirectional keys". Scyther, however, models unidirectional keys by default. We approximate bidirectional keys by adding helper protocols that allow the adversary to replace K_{AB} by K_{BA} in terms. An example for a helper protocol can be found in the input model of 2-4-a, in Figure 7 on page 46.

Key Derivation Functions (KDF): In many protocols, keying material is transported and then input to a key derivation function to compute the final shared key. Suggestions of different key derivation functions are made by the standard in Part 2 ([9], Annex C.2) and Part 3 ([10], Annex B). The described functions are hash functions at least taking the exchanged secret as input. In most of them, additional parameters are input to the function. Examples for this are an integer denoting how long the derived key will be or an "algorith-

mID" saying for which algorithms the derived secret keying material will be used. Since we do not have a notion of how long the messages are and do not use the keys in algorithms, we omit these. As a variation, a key derivation function has been analyzed which takes as additional inputs the distinguishing identifiers of the two entities establishing the key. In some of the example KDFs it is said that information about the entities can (in some cases optionally) be input and this should at least include their identifiers.
If two models are necessary for modeling both variants, the name of the one which only takes the keying material as input ends with *"1"*; the model also taking the identifiers as input with *"2"*.

Certificates: In some of the protocols, an entity sends its certificate for the public key to a receiver which obtains from this the public key of the sender. We assume pre-distributed certificates in the default Scyther setup, and model certificate parts of messages by the public keys. For the purpose of consistency, the sent public key is defined using a macro, specified as $Cert_I$ if it is the certificate of the entity I.

Authenticated Channels: In the protocols 3-PKT-1 and 3-PKT-2, messages are transmitted over authenticated channels which include, according to the standard, data origin authentication, as well as data integrity. In the protocol 3-PKT-3, the message is said to be transmitted *"in an authenticated way"*. All these authenticated channels are modeled in Scyther by making an approximation. A secret and a public function are defined, sk1 and pk1 respectively, and they are defined to be the inverse of each other. Then, the initiator encrypts the message with the secret function that can be decrypted by the receiver, what therefore models an authenticated channel. We assume a public key infrastructure where, in contrast to the predefined secret keys, an adversary that is capable of revealing secret keys can not reveal these new defined ones.

Key Agreement Protocols: In the key agreement protocols 3-KA-1, 3-KA-2, 3-KA-3, 3-KA-4, 3-KA-5 and 3-KA-7, the standard provides an abstract description of the function used and says at the end for example that the presented protocol is *"an example of Diffie-Hellmann"*. We based the models on the Diffie-Hellmann protocol as follows: The common element g appearing in these protocols, corresponds to the common group element in Diffie-Hellmann. Often, it is described that an entity randomly and secretly generates an r and inputs this, together with the g to the function F; it computes *F(g,r)*. The result of this function is then sent to the partner. We interpret this as g^x in Diffie-Hellmann, which is in turn modeled with a hash function exp(g,x) in Scyther.
In all the protocols, the functions F are modeled as exponentiation and helper protocols are added to model their commutativity.

Private and Public Key Agreement Keys: Most of the key agreement protocols contain a private and a public key agreement key h_x and p_x, where $p_x = F(h_x, g)$ and x denotes the entity. We model the private key as the secret key of the entity. Additionally, we model the public key agreement key p_x as $exp(g, sk(X))$ and its pre-distribution by a helper function from which an adversary can learn this term.

MQV Key Agreement Protocols: The protocols 3-KA-8, 3-KA-9 and 3-KA-10 use elliptic curve cryptography and are explained to be examples of MQV. We base our models on the models "HMQV-twopass" and "HMQV-C" by Cas Cremers, which can be found in the directory *Protocols/AdversaryModels* when downloading Scyther from [5]. As these original models are based on Diffie-Hellmann, we had to change them to be closer to the standard which is based on elliptic curves.

3.3 Security Properties

The security properties claimed by the standard consist of entity authentication, key authentication and secrecy of the key. All the protocols, except for the public key transport protocols of Part 3, have the goal to establish or agree upon a new session key. Therefore, the claim of session key (*SKR* in Scyther) is analyzed in all of them, which includes the test of secrecy. In terms of authentication, the standard does not specify what kind of authentication is provided. We, therefore, analyze four standard authentication properties in Scyther, which are aliveness, weak agreement, non-injective agreement and non-injective synchronization.
We recall the informal definitions of aliveness, weak agreement and non-injective agreement from Lowe's paper *A hierarchy of authentication specifications* [8] and refer to [6] for formal definitions.

Aliveness [8]: *"We say that a protocol guarantees to an initiator A* aliveness *of another agent B if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol."*

Weak Agreement [8]: *"We say that a protocol guarantees to an initiator A* weak agreement *with another agent B if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A."*

Non-injective agreement [8]: *"We say that a protocol guarantees to an initiator A* non-injective agreement *with a responder B on a set of data items ds (where ds is a set of free variables appearing in the protocol description) if, whenever A (acting as a initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was acting as responder in his run, and the two agents agreed on the data values corresponding to all the variables in ds."*

An informal definition for non-injective synchronization is recalled from [6]:

Non-injective Synchronization [6]: Non-injective synchronization *"ensures that the protocol is executed exactly as it would be if no adversary were present."* It is also said that this is only true if a single run of the protocol is executed, since replaying messages between sessions is still possible.

The standard considers implicit key authentication, key confirmation and explicit key authentication. Since **implicit key authentication from an entity A to an entity B** is defined in the standard [10] as being *"the assurance for entity B that entity A is the only other entity that can possibly be in possession of the correct key"*, this is analyzed with the claim for session key in entity B. If an entity B can be sure that the key is secret between him and A, then he can also be sure that only A can possibly be in possession of it. **Key confirmation from A to B** is ([10]) *"the assurance for entity B that entity A is in possession of the correct key"*. This fact is analyzed by examine if an entity B can commit to the value of the key in A. If this is possible, he can be sure that A is indeed in possession of the correct key. Finally, **explicit key authentication** is in [10] described to be the assurance that the other entity is the only one in possession of the correct key and was therefore analyzed by combining implicit key authentication and key confirmation.

4 Presentation of Models

In this section, we present the models that we developed and which can be found in [1]. The first section presents overview tables of the models, which show what mechanisms are used by the protocols and what the standard claims about them. In the following sections, we give for each model a short description and explain the variants. If additional claims, not appearing in the tables, or requirements are mentioned in the standard, they are also explained. For a more detailed description of the protocols, we refer to the standard ([9], [10]).

4.1 Overview Models

An overview of the models can be found in the tables 1 and 2. After the name in the first column, the second column shows how many messages are exchanged in the protocol.
In Table 1, the third column denotes what mechanism is used and means by

PtP that the protocol is a point-to-point key establishment, which means that only the entities that wish to establish a key are communicating and no third party is involved.

KDC that the protocol uses a key derivation center as third party which generates/acquires keys and distributes them to the entities that wish to establish a key.

KTC that the protocol uses a a key translation center as third party which does not generate keys itself, but rather transfers keys generated by one of the entities to the another entity where both share a key with the KTC.

In the next columns the positive and negative statements of the standard about what is achieved by the protocol are listed. EA denotes in the positive claims that mutual authentication is given and in the negative that no authentication is given. For unidirectional claims, EA_{XY} denotes entity authentication from X to Y, where X and Y are substituted by I, denoting the initiator or R, denoting the responder (in some cases also the third party involved, that is the KDC or KTC). KA denotes that key authentication is claimed and "-" that no negative/positive statements are made. SA/REF denote that substitution

attacks/reflection attacks are possible; the protocol is only secure in an environment where these attacks are not possible.
Table 2 on the next page shows all the protocols of Part 3 of the standard. Because the type of mechanism used can be seen in the name, this column is omitted. The notation remains, but new notions for key authentication are distinguished. IKA_{XY} denotes implicit key authentication from X to Y or mutual, if no entities are attached. Similarly, EKA_{XY} is explicit key authentication from X to Y. SEC is the claim that the key is secret and KC_{XY} stands for key confirmation from X to Y. Finally, REP means that replaying attacks are possible. Joint key control is also a claim appearing in the standard, which is not included in the table because it was not analyzed.

Model	**Passes**	**Mech.**	**Positive Claims Standard**	**Negative Claims Standard**
2-1	1	PtP	-	no EA,no KA
2-2	1	PtP	-	no EA,no KA
2-3-a	1	PtP	EA_{IR},no SA	-
2-3-b	1	PtP	-	SA
2-4-a	2	PtP	EA_{RI}	-
2-4-b	2	PtP	-	SA
2-5-a	2	PtP	EA	-
2-5-b	2	PtP	-	SA
2-6-a	3	PtP	EA	-
2-6-b	3	PtP	-	REF
2-6-v	4	PtP	-	-
2-7	3	KDC	-	no KA
2-8-a	4	KDC	EA	-
2-8-b	4	KDC	$EA_{I,KDC}$	-
2-8-c	3	KDC	-	no EA
2-9-a	5	KDC	EA	-
2-9-b	4	KDC	-	no EA
2-10	3	KDC	$EA_{I,KDC/KDC,I/KDC,R}$	-
2-11	3	KTC	-	-
2-12-a	4	KTC	EA	-
2-12-b	3	KTC	-	no EA
2-13-a	5	KTC	EA	-
2-13-b	4	KTC	-	no EA

Table 1: Overview Models Part 2

4.2 Symmetric Point-to-Point Mechanisms

Isoiec-11770-2-1: In this protocol, the initiator sends a time variant parameter, TVP_I, to the receiver. The key is derived on both parties' sides by a key derivation function of the already shared key K_{AB} and the sent TVP.

Isoiec-11770-2-2: In the second mechanism, the key is supplied by the initiator, which then sends the keying material encrypted with an already shared

Model	Passes	Positive Claims Standard	Negative Claims Standard
3-KA-1	0	IKA	no KC
3-KA-2	1	IKA_{RI}	no KC
3-KA-3-a	1	IKA,EKA_{IR},EA_{IR},KC_{IR}	-
3-KA-3-b	1	IKA,EKA_{IR},EA_{IR},KC_{IR}	$REP\ key$
3-KA-4	2	-	no KA,no EA,no KC
3-KA-5-a	2	IKA	-
3-KA-5-b	2	EKA_{RI},KC_{RI}	-
3-KA-6-a	2	SEC,IKA,EKA_{RI}	-
3-KA-6-b	2	SEC,IKA,EKA_{RI}	-
3-KA-6-c	2	SEC,IKA,EKA_{RI},KC_{RI}	-
3-KA-7-a	3	KA,EA,KC	-
3-KA-7-b	3	KA,EA,KC	-
3-KA-8	1	IKA	-
3-KA-9	2	IKA	-
3-KA-10	3	EKA	-
3-KA-11	4	EKA	-
3-KT-1-a	1	IKA_{RI},no REP	no KC,no EA_{IR}
3-KT-1-b	1	-	REP
3-KT-2-a	1	EKA_{IR},IKA_{RI},no $REP\ key$	$REP\ BE$
3-KT-2-b	1	-	no EA_{IR}
3-KT-2-d	1	no $REP\ BE$	-
3-KT-2-c	2	EA	-
3-KT-3-a	1	EA_{IR},no $REP\ key$,KC_{IR}	-
3-KT-3-b	1	-	no EA
3-KT-4-a	2	EA_{RI},KC_{RI},$IKA_{m}athitIR$	-
3-KT-4-b	2	-	-
3-KT-4-c	4	EA	-
3-KT-5-a	3	EA,IKA_{RI},KC_{IR}	-
3-KT-5-b	3	-	-
3-KT-6-a	3	EA,KC,IKA_{RI}	-
3-KT-6-b	3	-	-
3-PKT-1	1	-	-
3-PKT-2-a	2	-	-
3-PKT-2-b	2	-	-
3-PKT-3	1	-	-

Table 2: Overview Models Part 3

key K_{AB} to the receiver.

Isoiec-11770-2-3: 2-3-a is the point-to-point key establishment protocol as it is originally proposed in the description, with all optional parts included. The key K is supplied by the initiator which then sends it to the receiver. In the variation 2-3-b of the protocol the identifier of the receiver in the sent message is omitted.

Isoiec-11770-2-4: This protocol consists of two messages. The initiator sends a random number to the other entity, which then replies with a message encrypted with K_{AB} and consisting of the same random number, the distinguishing identifier of the initiator and keying material. Apart from what can be seen in Table 1, the standard claims that uniqueness/timeliness is controlled by the random number.
The variant 2-4-a includes the optional parts, while 2-4-b omits the distinguishing identifier of the initiator in the second message.

Isoiec-11770-2-5: In this two message protocol the initiator first sends the message consisting of a time stamp/sequence number T_A/N_A, the distinguishing identifier of the receiver and the keying material F_A encrypted with K_{AB} to the responder. The response has exactly the same form. The standard claims that uniqueness/timeliness is controlled by T/N.
2-5-a models the protocol as it is originally proposed and the alternative 2-5-b models that the distinguishing identifiers of both parties can be left away.

Isoiec-11770-2-6: 2-6-a is the protocol with the optional parts that consists of three messages between two parties that can both contribute part of the established key. The standard claims that uniqueness/timeliness is controlled by the random numbers.
For protocol 2-6-b, several possibilities arising from the *NOTE 1* and *NOTE 2* in the standard (p.9) [9] have been analyzed. In *NOTE 1* the standard explains that one of the keying material fields F_A or F_B can be left empty. If however the optional distinguishing identifier is left away, the protocol is only secure in environments where reflection attacks are not possible according to the standard. In 2-6-b-1 F_A is omitted, but the identifier I_B is included, in 2-6-b-2 F_B is omitted but the identifier I_B is included, in 2-6-b-3 both F_A and I_B are omitted and finally in 2-6-b-4 F_A and F_B are both included while I_B is omitted.

Isoiec-11770-2-6-v: This variation of the protocol 2-6 is mentioned in *NOTE 3* [9] and is constructed from two parallel instances of the key establishment mechanism 2-4, one started by each entity. No statements are made whether or not the originally claimed properties still hold.

4.3 Symmetric Mechanisms using a Key Derivation Center

Isoiec-11770-2-7: The initiator first sends the identifier of the entity he wishes to communicate with to the KDC. The KDC replies to him with a message consisting of two parts, one encrypted for him and one for the intended partner which the initiator can forward. No variation of this protocol is described.

Isoiec-11770-2-8: In this four message key establishment, the initiator first sends a message consisting of a TVP and the distinguishing identifier of the intended partner to the KDC. Then, he gets a replay from the KDC consisting of two parts which both include the session key: the first part is encrypted for the initiator, the second for the responder. The initiator forwards the second part of the message together with an optional new part that is encrypted with the new session key. The responder optionally send a message back, also encrypted with the new session key. The standard makes the claim that uniqueness/timeliness is controlled by the T/N.
The protocol with the optional parts in brackets is modeled in 2-8-a. In *NOTE 3* [9], the standard claims that by the inclusion of a MAC over TVP_A with a shared secret key of the initiator and the KDC, authentication of the requesting entity by the KDC is provided. This variation is modeled in 2-8-b. The last variant of the protocol is to leave out the optional parts, which was analyzed in the model 2-8-c.

Isoiec-11770-2-9: The initiator first sends a random number R to the responder. This forwards the received together with a new random number generated by him and the distinguish identifier of the initiator to the KDC. The responder gets a replay from the KDC which consists of two parts, one encrypted for him and one for the initiator, which both contain the keying material. He forwards the part for the initiator, optionally with a message encrypted with the new key. Also optionally, he gets a replay again encrypted with the new key. The standard makes the claims that in this protocol uniqueness/timeliness is controlled by the random numbers. 2-9-a is the protocol with the optional parts, while in 2-9-b the optional messages are left away.

Isoiec-11770-2-10: In the first message, the initiator sends a T/N and the distinguishing identifier of the entity he wishes to establish a key with encrypted with their shared key to the KDC. This replies to the initiator with a similar message as he sends directly to the intended receiver. The standard claims that this protocol achieves mutual authentication between the initializing entity and the KDC, as well as unilateral authentication of the KDC to the receiver of the third message. No statements about the authentication between the communication partners are made. Further, uniqueness/timeliness is claimed to be controlled by the time stamps or sequence numbers.

Key Derivation Function (KDF) Input: In the protocols 2-8 and 2-9, a finer distinction is made by the choice of what to input to the KDF. This gives raise to the protocol versions "1" and "2" in 2-8-a, 2-8-b and 2-9-a. As in 2-8-c and 2-9-b the only difference is in the claims, both versions can be analyzed in the same model.

4.4 Symmetric Mechanisms using a Key Translation Center

Isoiec-11770-2-11: In the first message of this protocol, the initiator sends the distinguishing identifier of the entity he wishes to establish a key with and the keying material F provided by himself, encrypted with their shared key to the KTC. The KTC replies with a message encrypted with the shared key of the KTC and the receiver, which the initiator can forward to the receiver. No variations are described for this protocol.

Isoiec-11770-2-12: Message 1, which is sent by the initiator to the KTC contains the number one, a T/N, the identifier of the responder and the keying material, encrypted with the key shared between the two. The KTC responds with a message consisting of two parts, the first is encrypted with the same key, the second with the shared key between the KTC and the responder. Each part is again denoted by the message number, two and three respectively. After the responder gets the second part forwarded from the initiator, there are optionally two more messages between the responder and the initiator which are both encrypted with the new shared key.

Isoiec-11770-2-13: In this protocol, the initiator first sends a random number R_I to the responder, which sends to the KTC the following message encrypted with their shared key: a random number R_R generated by him, the random number he just received from the initiator, the identifier of the initiator and the keying material. The KTC responds with the message consisting of two parts, one encrypted for each of the entities communicating. The responder then forwards in the third message the part encrypted for the initiator and optionally adds as second part a message with a new random number R'_R generated by him and the random number R_I, encrypted by the new session key. Also optionally, the initiator responds with R_I and R'_R encrypted with the new key.

Variations on Isoiec-11770-2-12, Isoiec-11770-2-13: 2-12-a and 2-13-a are the protocols with the optional parts, and 2-12-b and 2-13-b are the protocols which omit the optional messages. As in the protocols using a KDC (Section 4.3), to model both variations on what to input to the key derivation function KDF, two versions "1" and "2" are needed in 2-12-a and 2-13-a.

4.5 Asymmetric Key Agreement Mechanisms

A description of how we modeled the function F, as well as the private and public key agreement key, can be found in Section 3.2.

Isoiec-11770-3-KA-1: In this protocol, no messages are exchanged. Both entities compute the key with their private and the public key agreement key of the partner.

Isoiec-11770-3-KA-2: The initiator inputs to a function F a common element g and a random generated r and sends this to the receiver. The initiator computes the key with this r and the public key agreement of the receiver; the receiver computes the same key with the received function and his secret key. The standard says that since the receiver gets the message from a non-authenticated initiator, he should only use the key for functions that do not require trust in this entity's authentication.

Isoiec-11770-3-KA-3: The initiator first computes the new key with a random generated r and the public key agreement key of the receiver. He then

sends a message which includes a function of the random generated r. Also, he includes a MAC computed with the new key, which is signed by the initiator. The receiver then computes the key from the received function of r and its private key agreement key.
In 3-KA-3-a, the optional TVP is included, where the standard makes the additional claim that this prevents a replay of the key token. 3-KA-3-b is the model where this optional part is omitted.

Isoiec-11770-3-KA-4: This mechanism does not require prior exchange of information. Both entities randomly generate a r and send the function of it and a common element g to the partner. Both compute the key as their random generated key and the value they received.

Isoiec-11770-3-KA-5: In this protocol, the participating entities have agreed on a common one-way function w.
Both send the function of the common element g and a random generated r to the partner and compute the key as as the one way function of their own private and public key agreement key, the public information of the partner and the received function.

Isoiec-11770-3-KA-6: In this protocol, the initiator first sends the key token consisting of a random number r_A and a text to the responder. The responder signs a data block, called BS, which includes the old random number r_A and a newly generated random number. Then, he encrypts this block, together with other information, with the public key of the initiator and sends it back. The standard says in Part 3 [10](p. 19) *"The shared secret key consists of all or parts of entity B's signature Σ contained in the signed block BS, used with a key derivation function"*. For all the variants of the protocol, we chose to model the key as the whole signed block BS, that is $I_A, r_A, r_B, \mathit{Text2}$ signed with sk_B.
3-KA-6-a is the protocol as originally proposed and 3-KA-6-b models that the text fields explicitly declared as optional are left away. No new assumptions are made on what should hold after this change.
Another variant, modeled in 3-KA-6-c, is to add a MAC in the text field sent within the public encryption of the sender, but outside the block BS. This variation is mentioned in *NOTE 4* [10].

Isoiec-11770-3-KA-7: First, the initiator sends the functions of a secretly generated r and a common element g to the responder. For messages 2 and 3 both entities construct a key token including the function with the random number generated by themselves, as well as the function they received from the partner.
3-KA-7-a models that in both messages 2 and 3, the key token is signed and a MAC with the new key is computed over it. 3-KA-7-b models the alternative mentioned in the description, namely that instead of the MAC the signed part is additionally encrypted with the new established secret key.

Isoiec-11770-3-KA-8: This protocol is based on elliptic curve cryptography. The initiator sends a function of a secretly generated r and a common element g to the receiver. The initiator computes the key as a key derivation function taking as input the random generated r, the private key agreement key and the public key agreement key of the receiver. The receiver inputs to the same function the received key token, its private and both public key agreement keys. This mechanism is an example of MQV and we based it on the model "HMQV-twopass" which is provided when downloading the Scyther tool from [5] under "AdversaryModels".

Isoiec-11770-3-KA-9: In this protocol, both entities compute a function of a common element g and a randomly generated r and send this to the partner. Then, they input to a KDF their own random number, the message they received, their private key agreement key and the public key agreement key of the partner. The mechanism is an example of MQV and we based it on the model "HMQV-twopass" which is provided when downloading the Scyther tool from [5] under "AdversaryModels".

Isoiec-11770-3-KA-10: Both entities compute a function based on a common element g and a randomly generated r. In the first message, the initiator sends this to the other entity. The responder answers with his key token, together with a MAC over this and the received key token. The initiator answers with the same MAC. To distinguish the two messages, additionally the message number (two and three) is input to the MAC. It is explained that this is an example of MQV, so we based it on the model "HMQV-C", provided when downloading the Scyther tool from [5] under "AdversaryModels".

Isoiec-11770-3-KA-11: In this protocol, the initiator sends a random integer r_I to the receiver. This replies with a random integer r_R chosen by him, together with his certificate. The initiator then generates a new random integer r'_I and computes the key as $KDF(r_I, r_R, r'_I)$. He then constructs the key token KT_{A2} consisting of the new random integer encrypted with the public key of the receiver. Then, he sends KT_{A2} together with a MAC of the new key computed over the first message he sent and KT_{A2}. The receiver verifies everything and replies with the message consisting of a MAC over the second message he sent before, also using the newly computed key.

Requirements: In the protocols 3-KA-1, 3-KA-5, 3-KA-7, 3-KA-8, 3-KA-9 and 3-KA-10 the requirement is that both entities have a private and a public key agreement, as well as access to an authenticated copy of the public one of the partner. In 3-KA-2, only the receiver needs to have a private and public agreement key, and the initiator access to its public information.
In 3-KA-3 the initiator, in 3-KA-6 the responder is required to have an asymmetric signature system; the respective other entity has access to the public verification transformation.
In 3-KA-6 the initiator and in 3-KA-11 both entities, are required to have a asymmetric encipherment system; the respective other entity has access to the public encipherment transformation.

4.6 Asymmetric Key Transport Mechanisms

Isoiec-11770-3-KT-1: The initiator encrypts his distinguishing identifier, the key, a TVP and a text and sends it together with another text to the receiver. It is claimed that replaying the key token should be made impossible by the

TVP included.
The variant 3-KT-1-a of this protocol includes all optional parts and the variation 3-KT-1-b omits the optional *TVP* as well as the optional text within the encryption.

Isoiec-11770-3-KT-2: This key transport protocol transports a secret key enciphered and signed from an initiator to a receiver in one message.
In the variation 3-KT-2-a, all the parts from the message draft are included, but not the optional parts only mentioned in the *NOTES*. It is claimed that replays of the key token are prevented by the *TVP*, but the negative statement is made that an additional *TVP* would have to be included in the text field 1 in order to prevent a replay of the data block *BE*. The protocol 3-KT-2-b models that the optional *TVP*, as well as the first text are left away.
3-KT-2-v is modeling the remark in *NOTE 6* [10]. It is explained that the distinguishing identifier of the sender is included in the enciphered block *BE* to prevent the initiator from misappropriating an enciphered key block intended for use by another entity. This should then be achieved by comparing the identity with the identity's signature. 3-KT-2-d treats the case where an additional *TVP* is added in the *Text1* field. This variant is mentioned in *NOTE 4* [10] and should prevent replays of the key data block *BE*.

Isoiec-11770-3-KT-2-c: *NOTE 9* [10] suggests two combined executions of the mechanism 3-KT-2 and therefore gives rise to key transport protocol which now has two passes.

Isoiec-11770-3-KT-3: In this protocol, the initiator signs a message containing the key and the identifier of the receiver and encrypts this with the public key of the receiver.
In 3-KT-3-a, the protocol is modeled with all optional parts appearing in the description. The standard claims that the *TVP* provides entity authentication of the sender to the receiver and prevents a replay of the key token. The variant 3-KT-3-b treats the case where the *TVP* and the optional texts are left away.

Isoiec-11770-3-KT-4: In the first message, the initiator sends a random number r_I together with a text to the responder. The responder forms a data block, containing the key, encrypts this with the public key of the initiator, and signs this together with additional information.
3-KT-4-a is the protocol as it is originally proposed in the description. In the variant 3-KT-4-b, the case is modeled where the optional text fields, as well as the optional r_B (as mentioned in *NOTE 7*, [10]) are omitted. It is explained that the random number is only included to be consistent with another standard.

Isoiec-11770-3-KT-4-c: In *NOTE 8* [10] the standard suggests that two executions of the key transport mechanism 3-KT-4 are combined, which is modeled in this variant.

Isoiec-11770-3-KT-5: In this protocol, two keys are transported. In the first message the initiator sends a random number and a text to the responder. This, first generates the data block BE_1 containing the key K_R, which he encrypts with the public key of the initiator. He then signs it with additional information,

such as random numbers, and sends it back. The initiator generates the data block BE_2 consisting of exactly the same (with respect to his identity now) and also sends the same information signed back.
3-KT-5-a is the protocol as it is originally proposed. The variation 3-KT-5-b omits all optional parts, including the data block BE_1 since in the end of the description it is said that either BE_1 or BE_2 can be omitted if only unilateral key transport is required. We chose to model the transport of the key originating from the initializing party.

Isoiec-11770-3-KT-6: First, the initiator sends a data block containing the key that he provides, encrypted with the public key of the responder. The responder constructs a similar block which contains the keying material provided by him and sends this back, encrypted with the public key of the initiator. The initiator replies in the last message the random number he just received.
3-KT-6-a is the protocol as it is originally proposed and in the variation 3-KT-6-b, the optional text fields are omitted.

Requirements: The responder in 3-KT-1, 3-KT-2 and 3-KT-3, as well as the initiator in 3-KT-4, and both entities in 3-KT-5 and 3-KT-6, are required to have a asymmetric encipherment system; the partner entity has to have access to an authenticated copy of the public encipherment transformation.
The initiator in 3-KT-2 and 3-KT-3 , the responder in 3-KT-4 and both entities in 3-KT-5, are required to have an asymmetric signature system; the respective partner has to have access to the public verification transformation.

4.7 Asymmetric Public Key Transport Mechanisms

Public Key Information: In all the presented mechanisms, the public key information is explained to include at least the initiator's distinguishing identifier and its public key. The standard further explains that it may additionally contain a serial number, a validity period, a time stamp and other data elements. We chose to model the public key information as the distinguishing identifier, the public key, and a *TVP* (also see models in [1]).

Isoiec-11770-3-PKT-1: In this public key transport mechanism an initiator sends its public key information to a receiver over a channel that provides data origin authentication and data integrity.

Isoiec-11770-3-PKT-2: In this second public key transport mechanism, an unprotected channel is used for the first message, but the second message is transmitted over an authenticated channel.
3-PKT-2-a is the protocol as originally proposed, and in the variation 3-PKT-2-b, an additional signature by the initiator is added over the first message sent (as mentioned in *NOTE 3* [10]).

Isoiec-11770-3-PKT-3: In this protocol, the initiator sends its certificate to the receiver in an authenticated way. The certificate is modeled as the public key, as explained in the Section 3.2.

Claims PKT: In all public key transport mechanisms, no explicit statements on what is achieved are made.

5 Analysis Results with respect to a Dolev-Yao adversary

In this section we present the attacks found with Scyther with respect to a Dolev-Yao adversary. For a discussion of other adversary models, we refer to Section 6.
First, an overview table is presented where it can be seen which attacks are found for each protocol and what claims are violated by it. Then, we group similar attacks together and explain them in the following sections. In some cases, one attack can be considered to be of more than one attack type, then this is mentioned where the attack is presented. In cases where it is not clear from the attack description, for clarifying in which role the mentioned attack has been falsified, a *(I)*, *(R)* or *(3rd)* is added if it is found in the claims of the initiator, the responder or the third party respectively; *(I,R)* denotes that both is true. When nothing else is said, authentication is always meant with respect to the entity one wants to establish a key with, we sometimes refer to this entity as *(communication) partner*.
In 5.11, we describe two attacks in full detail.

5.1 Attack Overview

The tables 3 to 8 provide the reader with an overview of the attacks. The first column contains the protocol, the second column what kind of attack was found. For the explanation of the attacks we refer to the annotated sections, and only list the shortcuts here:

ACP Absent Communication Partner (Section 5.3)

RepText Replacement of Unprotected Parts (Section 5.4)

RepUnr Replacement of Unreadable Messages (Section 5.5)

1En2Ro One Entity In Two Roles (Section 5.6)

SA Substitution Attack (Section 5.7)

RMU Role-Mixup Attack (Section 5.8)

UKS Unknown Key Sharing Attack (Section 5.9)

ConstSK Constructing Session Key (Section 5.10)

The column *in Claims* shows what claims do not hold, again with *I/R/3rd* for clarifying in what role the claim was made. *SKR* means that the claims for session key, which includes secrecy, is falsified. *auth* denotes that in this entity all claims of authentication are falsified, that is aliveness, weak agreement *(WA)*, non-injective agreement and non-injective synchronization. Since the following properties are often falsified together, *auth1* means that aliveness and weak agreement are falsified, and *auth2* is the same for non-injective agreement and non-injective synchronization. Finally, *CommK* denotes that the claim of

committing to the key of the partner is falsified, *CommE* denotes that the claim to commit to the other entity's identity is falsified, and *Comm* means both are true. Where nothing else is stated, *auth1*, *CommK* and *CommE* are meant with respect to the entity one wants to establish a key with.
The last column compares the results with the standard by listing for what claims a contradicting attack was found. When some kind of authentication is violated, but not all forms of it, the claim is written in brackets.
Tables 3, 4, and 5 show the results for the protocols of Part 2, Tables 6, 7 and 8 for the protocols of Part 3, sorted according to what mechanism they use.

Protocol	**Attacks**	**in Claims**	**Contrad.Std**
2-1	ACP (Section 5.3)	auth1(I)	
	ACP(Section 5.3)	auth,Comm(R)	
2-2	ACP (Section 5.3)	auth1(I)	
	RepText (Section 5.4)	auth2(R)	no*EA*
2-3-a	ACP (Section 5.3)	auth1(I)	
2-3-b	ACP (Section 5.3)	auth1(I)	
	SA,RMU (Section 5.7)	auth,Comm(R)	
2-4-a	ACP (Section 5.3)	auth,CommE(R)	
2-4-b	SA,RMU (Section 5.7)	auth,Comm(I)	
	ACP (Section 5.3)	auth,CommE(R)	
2-5-a	1En2Ro (Section 5.6)	auth2,Comm(I)	(*EA*)
	1En2Ro (Section 5.6)	auth2,CommK(R)	(*EA*)
2-5-b	SA (Section 5.7)	auth,Comm(I)	
	SA,RMU (Section 5.8)	auth2,CommK(R)	
2-6-a	-	-	
2-6-b-1	-	-	
2-6-b-2	-	-	
2-6-b-3	SA,RMU (Section 5.7)	auth,CommE(I)	
	SA,RMU (Section 5.7)	auth,Comm(R)	
2-6-b-4	SA,RMU (Section 5.7)	auth,Comm(I,R)	
2-6-v	RMU (Section 5.8)	auth2,Comm(I)	
	RMU (Section 5.8)	auth2,Comm(R)	

Table 3: Attacks found in Point-to-Point Mechanisms of Part 2

5.2 No attacks found

In almost all the protocols, there are no attacks found for the claims of session key in both the initiator's and the receiver's role. The only exception are the protocols 3-KA-2, 3-KA-4, 3-KA-11 and 3-KT-1-a/b.
In the following protocols and roles there are additionally no attacks found for aliveness, weak agreement, non-injective agreement, non-injective synchronization, as well as for the commitment to the key: 2-3-a(R), 2-4-a(I), 2-6-a(I,R), 2-6-b-1(I,R), 2-6-b-2(R), 3-KA-7-a/b(I,R), 3-KA-10(I,R), 3-KT-3-b(R), 3-PKT-2-b(R).

Protocol	Attacks	in Claims	Contrad.Std
2-7	ACP (Section 5.3)	auth(I)	
	RepUnr,RMU (Section 5.5,5.8)	auth(R)	
2-8-a-1	1En2Ro (Section 5.6)	Comm(I)	(EA)
	RepUnr (Section 5.5)	auth2(I,R)	(EA)
	UKS,RMU (Section 5.8,5.9)	auth1,Comm(R)	EA
2-8-a-2	1En2Ro (Section 5.6)	Comm(I)	(EA)
	RepUnr (Section 5.5)	auth2(I,R)	(EA)
2-8-b-1	1En2Ro (Section 5.6)	Comm(I)	
	RMU (Section 5.8)	Comm(R)	
	RepUnr (Section 5.5)	auth2(I,R)	
	RMU (Section 5.8)	WA(Initiator),(3P)	$(EA_{I,KDC})$
2-8-b-2	1En2Ro (Section 5.6)	Comm(I)	
	RepUnr (Section 5.5)	auth2(I,R)	
	RMU (Section 5.8)	WA(Initiator),(3P)	$(EA_{I,KDC})$
2-8-c	ACP(Section 5.3)	auth(I)	
	RMU (Section 5.8)	auth(R)	
2-9-a-1	RepUnr,RMU (Section 5.5)	auth2(I)	
	RMU (Section 5.8)	auth1(I),Comm(I,R)	EA
	UKS,RMU,SA (Section 5.8,5.9)	auth1(R)	EA
2-9-a-2	RepUnr,RMU (Section 5.5,5.8)	auth2(I)	(EA)
	RepUnr (Section 5.5)	auth2(R)	(EA)
2-9-b	RMU (Section 5.8)	auth(I)	
	ACP (Section 5.3)	auth(R)	
2-10	SA,RMU (Section 5.7,5.8)	auth1(of KDC), Comm(of KDC), auth2(I)	$EA_{KDC,I}$
	RMU (Section 5.8)	auth1(of KDC), Comm(of KDC)(R)	$EA_{KDC,R}$
	RMU (Section 5.8)	auth2(R)	
	RMU (Section 5.8)	auth1(Initiator) (3P)	$EA_{I,KDC}$

Table 4: Attacks found in Mechanisms of Part 2 using a KDC

5.3 Absent Communication Partner (ACP)

In the protocols 2-1, 2-2, 2-3-a/b, 3-KA-2, 3-KA-3-a/b, 3-KA-8, 3-KT-1-a/b, 3-KT-2-a/b/d/v, 3-KT-3-a/b, 3-PKT-1, 3-PKT-2-a/b and 3-PKT-3 due to the fact that there are no intended answers returning to the initiator, aliveness and weak agreement are not satisfied. Non-injective agreement and non-injective synchronization hold for all received messages, because there are none. In 3-KA-1, the same holds in both roles, since no messages are exchanged at all. In 2-7(I), 2-8-c(I), 2-9-b(R), 2-12-b(I) and 2-13-b(I) a similar attack is found. But since the entity has to communicate with the third party, additionally non-injective agreement and non-injective synchronization do not hold.
In the protocols 2-1, 2-4-a/b, 3-KA-2, 3-KA-5-b, 3-KA-6-a/b/c, 3-KA-8, 3-KT-1-a/b, 3-KT-4-a/b, all authentication claims as well as the commitment to the key are not satisfied in the receiving role, because the received message could have been sent by anybody. This is the case when the received message is not

Protocol	Attacks	in Claims	Contrad.Std
2-11	RepUnr (Section 5.5)	auth,Comm(I)	
	SA,RMU (Section 5.8)	auth1(R)	
2-12-a-1	1En2Ro (Section 5.6)	auth2,Comm(I)	(*EA*)
	RepUnr (Section 5.5)	auth2(I,R)	(*EA*)
2-12-a-2	RepUnr (Section 5.5)	auth2(I,R)	(*EA*)
2-12-b	ACP (Section 5.3)	auth(I)	
	RepUnr (Section 5.5)	auth2,CommK(R)	(no *EA*)
2-13-a-1	RepUnr (Section 5.5)	auth2(I,R)	
	UKS,RMU (Section 5.8,5.9)	auth1(R)	*EA*
	UKS,RMU,SA (Section 5.7,5.8,5.9)	auth1(I),Comm(I,R)	*EA*
2-13-a-2	RepUnr (Section 5.5)	auth2(I,R)	(*EA*)
2-13-b	ACP (Section 5.3)	auth(I)	
	RepUnr (Section 5.5)	Comm(R)	
	RMU (Section 5.8)	auth1(R)	

Table 5: Attacks found in Mechanisms of Part 2 using a KTC

encrypted or only encrypted with the public key of the receiver. The same is true for both entities in the protocols 3-KA-4, 3-KA-5-a and 3-KA-9.

5.4 Replacement of Unprotected Parts (RepText)

In the following protocols and roles, there are no attacks found for aliveness and weak agreement, and for the commitment to the key. Non-injective agreement and non-injective synchronization, however, are falsified by the attack that the unprotected text parts can be replaced by an adversary. This attack can be found in the claims of 2-2(R), 3-KA-6-a/b/c(I), 3-KT-2-a/d/v(R), 3-KT-3-a(R), 3-KT-4-a/b(I), 3-KT-5-a/b(I,R), 3-KT-6-b(R) and 3-PKT-2-a(R).
In 3-KA-5-b, 3-KT-4-c(I) and 3-KT-6-a(I), additionally the commitment to the key is falsified by this attack.

5.5 Replacement of Unreadable Messages (RepUnr)

In the protocol 2-11 the initiator has to forward a message from the third party to the receiver which he can not read. For this reason he will not notice if an adversary just replaces this message with something else. Therefore, the claims of authentication do not hold, and neither the commitment to the key. A similar observation is made in the protocol 2-12-b(R) where only part of the message can not be read, and only non-injective agreement, non-injective synchronization, and the commitment to the key are not satisfied. This attack results in the entities not agreeing on the key they share. Similarly, in 2-7(R) all the responder's claims of authentication do not hold and in 2-13-b(R) commitment to the key/identity, because the part that the initiator can not read is forwarded to the receiver from the third party directly. In 2-7(R) this attack additionally constitutes a role-mixup attack, because the third party gets the first message from the claiming entity in the other role.
The initiator in the protocols 2-8-a-1(I,R), 2-8-a-2(I,R), 2-8-b-1(I,R), 2-8-b-2(I,R), 2-9-a-1(R), 2-9-a-2(R), 2-12-a-1/2(I,R), 2-13-a-1(I,R) and 2-13-a-2(I,R)

Protocol	Attacks	in Claims	Contrad.Std
3-KA-1	ACP (Section 5.3)	auth1(I,R)	

3-KA-2	ACP (Section 5.3)
	ACP(Section 5.3)
	ConstSK (Section 5.10)
		auth1(I)
		auth,Comm(R)
		SKR(R)	

3-KA-3-a/b	ACP (Section 5.3)	auth1(I)	
3-KA-4	ACP (Section 5.3) ConstSK (Section 5.10)	auth,Comm(I,R) SKR(I,R)	
3-KA-5-a	ACP (Section 5.3)	auth,Comm(I,R)	
3-KA-5-b	RepText (Section 5.4) ACP (Section 5.3)	auth2,Comm(I) auth,Comm(R)	
3-KA-6-a/b/c	RepText (Section 5.4) ACP (Section 5.3)	auth2(I) auth,CommE(R)	
3-KA-7-a/b	-	-	
3-KA-8	ACP (Section 5.3) ACP(Section 5.3)	auth1(I) auth,Comm(R)	
3-KA-9	ACP (Section 5.3)	auth,Comm(I,R)	
3-KA-10	-	-	
3-KA-11	SA,RMU (Section 5.8) ConstSK (Section 5.10)	WA,auth2,CommK(I,R) SKR(R)	 *EKA*

Table 6: Attacks found in Key Agreement Mechanisms of Part 3

is also unable to read part of the message that is not encrypted for him. However, in these protocols an adversary still needs to forward this part to the receiver, as well as his answer back to the initiator in order to make the entities believe that the protocol finished. Therefore, only non-injective agreement and non-injective synchronization are falsified with this attack, but the commitment to the key of the respective other entity is possible in 2-8-a-2(R), 2-8-b-2(R), 2-9-a-2(R), 2-12-a-1/2 and 2-13-a-2. In 2-8-a-1, 2-8-a-2(I), 2-8-b-1, 2-8-b-2(I) and 2-13-a-1(I,R) also non-injective agreement and non-injective synchronization are falsified with this attack, but there are additional claims that do not hold because of other attacks.
The attacks found in 2-9-a-1(I) and 2-9-a-2(I) for non-injective agreement and non-injective synchronization also exploit the fact that part of the message can not be read. In this case, however, they also constitute role-mixup attacks. The message needed to finish the protocol is rerouted from another run, where the roles of the entities establishing a key are switched.

5.6 One entity in two roles (1En2Ro)

The following attacks were found in protocols where one entity plays both the initiator's and the responder's role, which is not excluded by the standard. Instead of getting the answer from the other role, the initiator gets his own message reflected by an adversary. This reflection attack violates non-injective agreement, non-injective synchronization, and the commitment to the key and the identity of the other role and can be observed in the protocols 2-5-a(I) and 2-12-a-1(I). The attack found in 2-5-a(R), which violates non-injective agreement, non-injective synchronization, and the commitment to the key, but not the

Protocol	**Attacks**	**in Claims**	**Contrad.Std**
3-KT-1-a/b	ACP (Section 5.3)	auth1(I)	
	ACP (Section 5.3)	auth,Comm(R)	REP(b)
	ConstSK (Section 5.10)	SKR(R)	
3-KT-2-a/d	ACP (Section 5.3)	auth1(I)	
	RepText (Section 5.4)	auth2(R)	
3-KT-2-b	ACP (Section 5.3)	auth1(I)	no EA_{IR}
3-KT-2-c	RMU (Section 5.8)	auth2,Comm(I,R)	(EA)
3-KT-3-a	ACP (Section 5.3)	auth1(I)	
	RepText (Section 5.4)	auth2(R)	(EA_{IR})
3-KT-3-b	ACP (Section 5.3)	auth1(I)	(no EA)
3-KT-4-a/b	RepText (Section 5.4)	auht2(I)	(EA_{RI})
	ACP (Section 5.3)	auth,CommE(R)	
3-KT-4-c	RepText (Section 5.4)	auth2,Comm(I)	(EA)
	RMU (Section 5.8)	auth2	(EA)
3-KT-5-a/b	RepText (Section 5.4)	auth2(I,R)	(EA)
3-KT-6-a	RepText (Section 5.4)	auth2,CommK(I)	(EA),KC
	RMU (Section 5.8)	CommK(R)	(EA),KC
3-KT-6-b	RepText (Section 5.4)	auth2(R)	
	RMU (Section 5.8)	auth2(I,R),CommK(R)	

Table 7: Attacks found in Key Transport Mechanisms of Part 3

Protocol	**Attacks**	**in Claims**	**Contrad.Std**
3-PKT-1	ACP (Section 5.3)	auth1(I)	
	RMU (Section 5.8)	WA,auth2(R)	
3-PKT-2-a	ACP (Section 5.3)	auth1(I)	
	RepText (Section 5.4)	auht2(R)	
3-PKT-2-b	ACP (Section 5.3)	auth1(I)	
3-PKT-3	ACP (Section 5.3)	auth1(I)	
	RMU (Section 5.8)	WA,auth2(R)	

Table 8: Attacks found in Public Key Transport Mechanisms of Part 3

commitment to the identity of the partner, is also possible when an entity wants to establish a protocol with himself. The difference is that messages have to be replayed over several protocol runs.
Another attack is found in 2-8-a-1(I), 2-8-a-2(I), 2-8-b-1(I) and 2-8-b-2(I) which violates the commitment to the key and identity of the entity one wants to establish a key with. In this case, since both roles are played by the same entity, the message encrypted with the new key at the end of the protocol can be reflected to the initiator; because the two messages look the same if the identities of the communication partners are equal. The message part unreadable for the initiator gets substituted with an intruder nonce.

5.7 Substitution Attacks (SA)

All authentication claims, including the commitment to the key are falsified in 2-3-b(R), 2-4-b(I), 2-6-b-3(I,R)and 2-6-b-4(I,R) by a substitution attack that

takes the message from another protocol run where the test run is present in the other role with different assumptions. The attack exploits the fact that, because of the symmetry of the key, the intended direction of the message can not be seen. A similar thing can be observed in 2-10(I), where the messages are rerouted from a run where the roles of the initiator and the third party are switched. This attack is violating the initiator's claims of aliveness and weak agreement with respect to the third party, non-injective agreement, non-injective synchronization, and the commitment to the third party's identity. Both cases additionally constitute a role-mixup attack, because the entities do not agree on who is playing what role.
In 2-5-b(I), an adversary can masquerade as the responder by reflecting the message sent.

5.8 Role-Mixup Attacks (RMU)

A role-mixup attack has the result that the participating entities do not agree on who is playing what role in the protocol.
In the claim of 2-8-c(R) and 2-9-b(I) all authentication claims are falsified, with a similar attack falsifying non-injective agreement and non-injective synchronization in 2-10(R) and 2-11(R)(there, it also falsifies the commitment to the key and the identity of the partner). A send from a protocol where the third party has other assumptions on who is playing the initiator and the responder's role, is replayed as another receive, what constitutes a role-mixup attack.
In 3-PKT-1(R) and 3-PKT-3(R), all the messages can be rerouted to the claiming entity from a partner that thinks he is talking to somebody else. Since the entities do not agree on who is playing the role of the responder, this constitutes a role-mixup attack falsifying weak agreement as well as non-injective agreement and non-injective synchronization.
In 2-6-v(I), 3-KT-2-c(I) and 3-KT-4-c(R) only non-injective agreement, non-injective synchronization, and, except for 3-KT-4-c, the commitment to the key/the identity of the partner are falsified with an attack that reroutes the messages from another protocol where the roles of the partners are switched. Only one entity is present in each run and a send of a message can be rerouted as a receive of another message; because of the symmetry the two messages look alike.
In 2-5-b(R) and 3-KT-2-c(R), the same is true (where the commitment in 2-5-b refers to the key part sent by the partner and the commitment to the identity of the partner is possible), but both entities have to be present in the protocol run where the message is rerouted from. This also applies in 2-6-v(R), although in this case only the commitment to the other entity's identity and the commitment to the key sent by himself are not satisfied. In 2-8-b-1(I) also the commitment to the key as well as to the responder's identity do not hold because of a similar attack, in this case the receive of message 4 is combined from the send of message 2 and 3 from a protocol run where the roles of initiator and receiver are switched.
In the protocol 3-KA-11, the same is possible. This also falsifies weak agreement, non-injective agreement and non-injective synchronization and the commitment to the key in both entities' claims. In 3-KA-11 and 2-5-b(R), the attack not only constitutes a role-mixup attack, but also a substitution attack.
There are role-mixup attacks found in 2-8-a-1(R), 2-9-a-1(I,R), 2-10(R), 2-11(R)

and 2-13-a-1(I,R) that violate aliveness, weak agreement and commitment to the key/identity with respect to the communication partner (in 2-11 only for aliveness and weak agreement), except for 2-10 where this is also true with respect to the third party. They all exploit the fact that because of symmetry the message sent to and from the third party can be sent in the other direction if the roles are switched. So in all examples, the third party is required to be able to act as a normal entity and this entity has then switched roles in another protocol run from where the messages are rerouted. In the claims 2-9-a-1(R), 2-11(R) and 2-13-a-1(I) of aliveness and weak agreement with respect to the partner, as well as in the commitment to the key/identity in 2-13-a-1(I,R), this also constitutes a substitution attack, because the adversary can masquerade as the respective partner. A similar attack is found in 2-13-b(R), only violating aliveness and weak agreement there.

In 3-KT-6-b(I,R) non-injective agreement and non-injective synchronization, as well as in 3-KT-6-b(R) commitment to both keys are falsified with an attack that reroutes the messages over several runs where the two participating entities both appear in both roles.

In 3-KT-6-a(R) only the commitment to the key provided by the other entity is falsified by an attack which takes messages from runs where the roles are switched.

5.9 Unknown Key Sharing Attacks (UKS)

An unknown key sharing attack results in an entity having wrong assumptions on who it is sharing a key with. This applies if another entity computes the same key but is not the intended partner. In 2-8-a-1(R), 2-9-a-1(I,R) and 2-13-a-1(I,R) the role-mixup attack described in Section 5.8 also constitutes an UKS attack, because the claiming entity shares the key at the end with the entity he believes to be in the third party.

5.10 Constructing Session Key (ConstSK)

In the claims of the responder of the protocols 3-KA-2, 3-KA-11 and 3-KT-1-a/b, the claims for session key, and hence secrecy do not hold. An adversary can construct the keying material himself and send it to the receiver as if it was coming from the initiator. The receiver then thinks to share a key that is actually constructed and known to the adversary.

In 3-KA-4 this is true in both entities: the adversary can send keying material to both entities that then compute the key such that the adversary also knows it. He achieves this by sending only the element g as the expected *F(g,x)*; it is not excluded that this function yields g as result. The key is then equal to the keying material the entity sent and hence known to the adversary, because the entities compute the key as *exp(g,a)*, where a is their respective secretly generated element and g the value they received.

5.11 Illustrative Examples

For two examples we want to provide more details by showing the protocol, the model input to Scyther and one of the attacks found. We chose the protocol 2-10, because it is one of the many examples for a role-mixup attack which makes use of the fact that a third party can play the role of a normal entity and of the fact that messages are symmetric because of the bidirectional keys. 3-KA-11 is interesting, because it does not fulfill the claimed secrecy of the established key.

5.11.1 Example 1: 2-10

1. A $\rightarrow$ P: $\{T_A/N_A \| I_B \| Text_1\}_{K_{AP}}$
2. P $\rightarrow$ A: $\{T_P/N_P \| F \| I_B \| Text_2\}_{K_{AP}}$
3. P $\rightarrow$ B: $\{T'_P/N'_P \| F \| I_A \| Text_3\}_{K_{BP}}$

Figure 2: Protocol Isoiec-11770-2-10, using a KDC to establish a session key

Figure 2 depicts the design of protocol 2-10, which uses a KDC to establish a session key. The initiator first sends a time stamp or sequence number, the distinguishing identifier of the entity he wants to establish a key with, and an optional text, all encrypted with a shared key, to the KDC. Then, the KDC sends to both the initiator and the wished partner the following message encrypted with the respective shared key: A time stamp or sequence number (a different one in the two messages), the keying material, the distinguishing identifier of the other entity and an optional text.
As presented in Section 4.1, the standard claims that this protocol achieves mutual authentication between the initiator and the KDC, as well as unilateral authentication of the KDC to the receiver. All of these claims are shown to not hold by the analysis. We want to show the attack violating the weakest claim of authentication from the KDC to the receiver. (The general idea of this attack, as well as similar attacks, are given in Section 5.8.)
In Figure 3 we show the Message Sequence Chart of the the attack. Since the message is symmetric, an adversary can reroute a message from Bob in the role of the third party to Bob in the role of the receiver. This scenario is not excluded by the standard. Bob then thinks that he got the message from Alice in the role of the KDC, which was never alive. This hence contradicts the standard's claim that the receiver can authenticate the KDC. A fix for this problem can be found in Section 8.3.

5.11.2 Example 2: 3-KA-11

The protocol 3-KA-11 consists of the four passes shown in Figure 4 which are also explained in Section 4.5. The standard [10] says that the following is required: each entity must have an asymmetric encipherment system and each entity has access to an authenticated copy of the public verification transformation of the other entity. Additionally, they have agreed on a common key derivation function. The standard claims that the protocol achieves mutual explicit key authentication. For this to hold, implicit key authentication as well as key confirmation have to be given. Nevertheless, we show an attack that violates secrecy of the key in the claim of the responder. This means that implicit key authentication is not given.
As illustrated in Figure 5, an adversary can send random numbers generated by him to the responder Bob. Since the key consists of the random numbers input to a publicly known key derivation function, the adversary can obtain

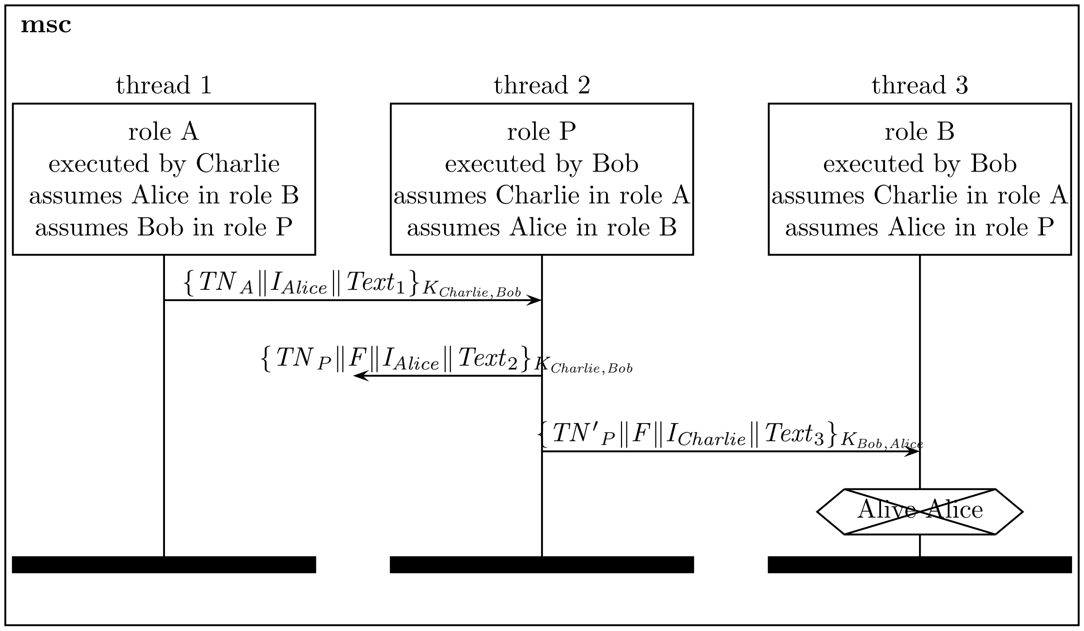

msc
thread 1
thread 2
thread 3
role A
executed by Charlie
assumes Alice in role B
assumes Bob in role P
role P
executed by Bob
assumes Charlie in role A
assumes Alice in role B
role B
executed by Bob
assumes Charlie in role A
assumes Alice in role P
$\{TN_A \| I_{Alice} \| Text_1\}_{K_{Charlie,Bob}}$
$\{TN_P \| F \| I_{Alice} \| Text_2\}_{K_{Charlie,Bob}}$
$\{TN'_P \| F \| I_{Charlie} \| Text_3\}_{K_{Bob,Alice}}$
Alive Alice

Figure 3: Attack on the protocol 2-10, where the third party can also be in the role of a normal entity. Bob wrongly concludes that Alice (as the KDC) is alive.

the key K_{AB} and therefore send the proper third message with a MAC that takes this key as input. The responder believes to share a secret key with Alice, even though Alice was never alive and the key is known to the adversary. This contradicts implicit key authentication from the initiator to the responder, which in turn contradicts the claim of the standard that mutual explicit key authentication is given.

6 Hierarchy of models with respect to Secrecy

The discussion in Section 5 was referring to a Dolev-Yao adversary model, that has the capability to derive the long term keys of dishonest agents before the

1. A → B: M_1
2. B → A: M_2
3. A → B: $KT_{A2} \| MAC_{K_{AB}}(M_1 \| KT_{A2})$
4. B → A: $MAC_{K_{AB}}(M_2)$

where
$M_1 = (r_A \| Text_1)$
$M_2 = (r_B \| Cert_B \| Text_2)$
$K_{AB} = kdf(r_A, r_B, r'_A)$
$KT_{A2} = \{r'_A\}_{pk_B}$

Figure 4: Protocol Isoiec-11770-3-KA-11 using a MAC for authentication

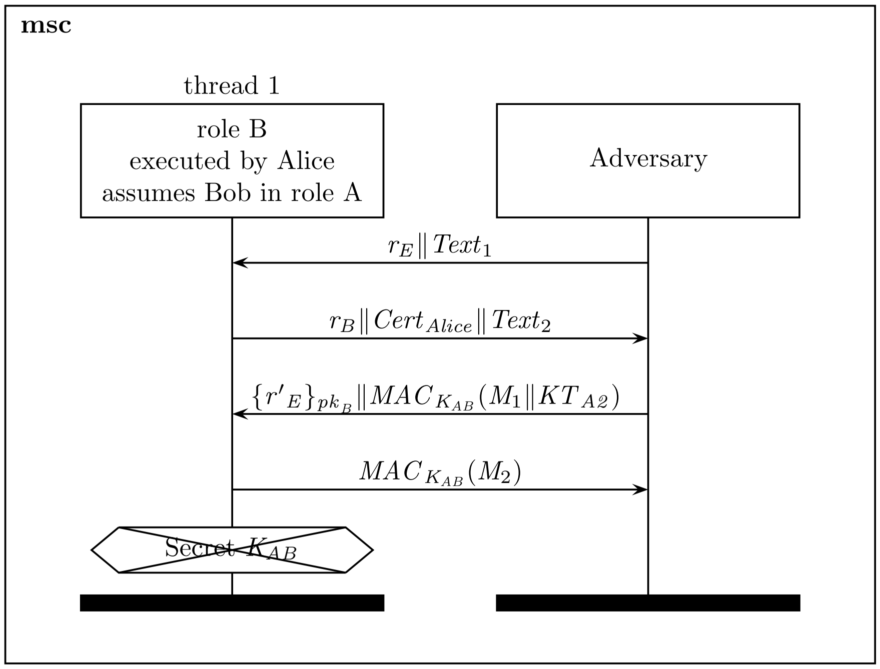
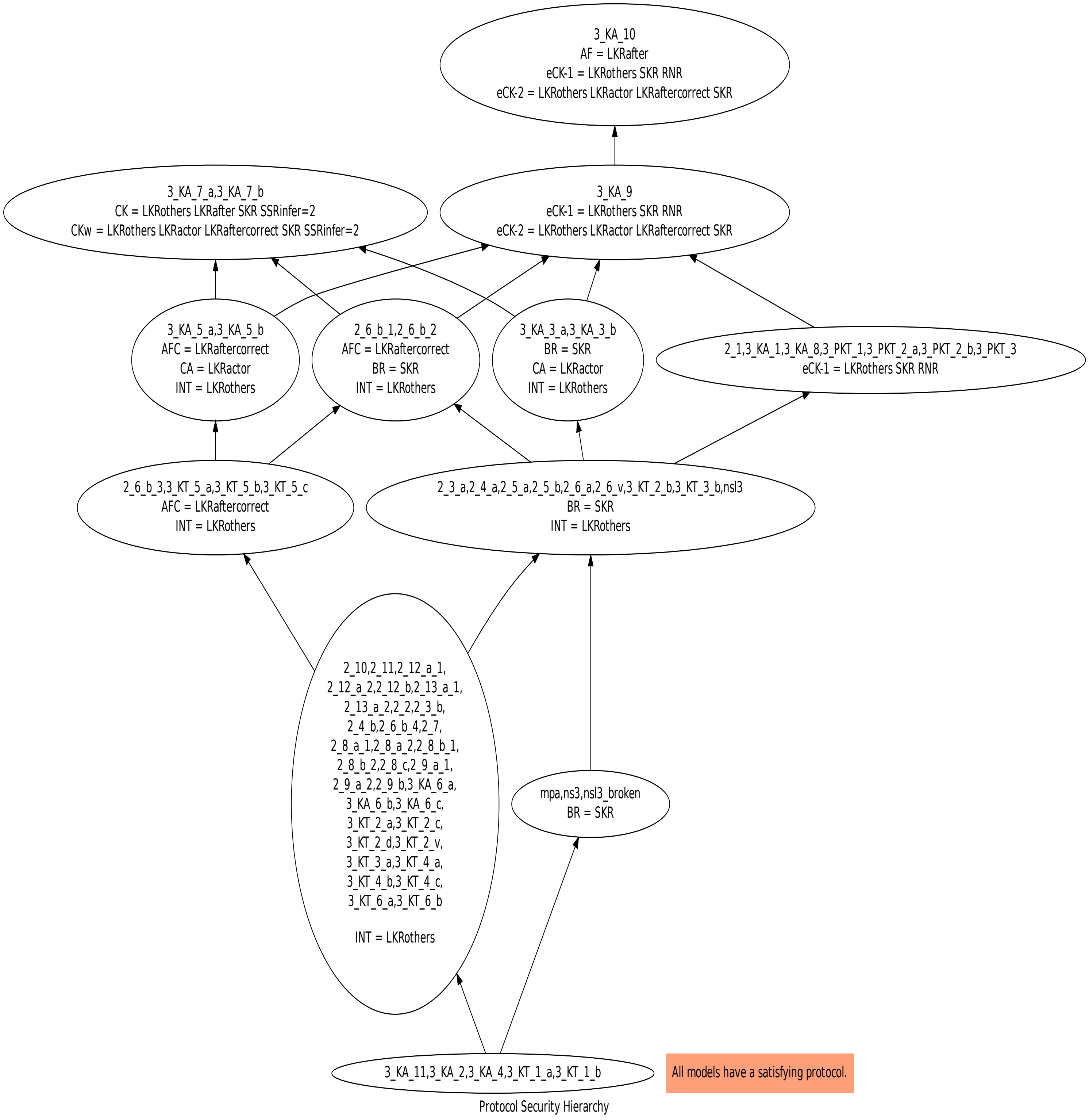


Figure 5: Attack on the protocol 3-KA-11, where the adversary can send intruder nonces as the expected random numbers and then derive the session key to also send the MAC to the responder. Alice believes to share a key with Bob that she actually shares with the intruder.

Figure 6: Protocol Security Hierarchy



Protocol Security Hierarchy

honest agents start their runs. Scyther also provides the option to check protocols with respect to different adversary-compromise rules. Each combination of the rules enabled describes a possible adversary model which can be analyzed. Additionally, it is possible to automatically generate a hierarchy which groups the models together according to the adversary model that they are secure in. Because of time limitations, we only consider the hierarchy with respect to the security property *secrecy* (see Figure 6). In this manner, we examine under what adversary capabilities the protocols provide secrecy of the established session key. Also, we want to find out what attacks are found if the capabilities are added that are possible a layer higher in the hierarchy. As in Section 5, *(I)* and *(R)* are added to point out in whose claims this attack is found.
First, we present the adversary-compromise rules and the tested adversary models that are analyzed with Scyther, then we present the different groups found in the hierarchy. At the end we discuss the special case of the public key transport mechanisms and draw a conclusion.

6.1 Adversary-compromise rules

We only want to give a short overview of the adversary-compromise rules analyzed, and refer to [2] or [3] for formal definitions. All possible sets of adversary-rules provide a new possible adversary model. The authors of [2] and [3] define the rules along the three dimensions *which* kind of, *whose* and *when* the data is compromised.

Compromise of long-term keys
It is explained in [3], that with the long-term key reveal rule LKR_{others} *"an adversary can learn the long-term keys of any agent a that is not an intended partner of the test run."* In the analysis of Section 5 the adversary was only capable of this rule.
The LKR_{actor} rule [3] *"allows the adversary to learn the long-term key of the agent executing the test run"*, which is also called the actor.
Finally, it is defined in [3] that *"The LKR_{after} and $LKR_{aftercorrect}$ rules restrict when the compromise may occur."* That is they allow the compromise of the long term key only after the run under test has finished. While this is the only premise of LKR_{after}, $LKR_{aftercorrect}$ *"has the additional premise that a finished partner run must exist for the test run."* If a protocol is secure against an adversary capable of LKR_{after} or $LKR_{aftercorrect}$, it is said to satisfy Perfect Forward Secrecy or weak Perfect Forward Secrecy respectively.

Compromise of short-term data
The session-key reveal event $SKR(tid)$ and state reveal event $SR(tid)$ indicate according to [3] that *"the adversary gains access to the session key or, respectively the local state of the run tid."* In this analysis we do not consider the SR rule further.
The random number reveal RNR [3] *"indicates that the adversary learns the random numbers generated in the run tid."* It is further explained in which run the adversary can compromise these things [3]: *"The rules SKR and SR allow for the compromise of session keys and the contents of a entity's local state. Their premise is that the compromised entity is not a partner run. In contrast, the premise of the RNR rule allows for the compromise of all runs, including*

the partner runs."

6.2 Adversary Models Considered in Hierarchy

As in [2], we consider seven adversary models for the hierarchy, each one consisting of another combination of the introduced adversary-compromise rules. In [2] it is explained on what models from the literature they are based on and what model contains which rules. As the allowed adversary rules are listed in each group in Figure 6, we omit these here.

6.3 Protocols not satisfying secrecy

As already seen in Section 5, the protocols 3-KA-2, 3-KA-4, 3-KA-11 and 3-KT-1-a/b do not provide secrecy of the established session key given the LKR_{others} rule. Consequently, even without a reveal they are insecure.

6.4 Secrecy given under LKRothers

The majority of protocols is found to preserve secrecy of the key given an adversary only being capable of LKR_{others}. The fact that they fulfill secrecy in this environments is exactly what was already found in the analysis of Section 5. Now we also know that they are not secure if any other compromising rule is added. We provide an overview of what attacks are found if SKR or $LKR_{aftercorrect}$, which still hold in the next stronger groups, are added.
Attacks after adding $LKR_{aftercorrect}$:
In the following protocols, the keying material F is directly sent in a message encrypted with the long-term shared key: 2-2, 2-3-b, 2-4-b(I), 2-6-b-4(R), 2-7(I,R), 2-8-a/b-1/2(I), 2-8-c(I,R), 2-9-a-1/2(I,R), 2-9-b(I), 2-10(I), 2-11(I,R), 2-12-a-1/2(I), 2-13-a-1/2(I,R), 2-13-b(I). The adversary can therefore decrypt the message and get the keying material by revealing the long term key after the session.
In 3-KA-6-a/b/c(I,R) and 3-KT-6-a/b(I,R), the adversary can do the same by revealing the secret key of one party.
In the protocols 3-KT-2-a/d/v(I,R), 3-KT-2-c(I), 3-KT-4-a/b(I), 3-KT-4-c(R) the adversary can first decrypt a signature, because he has access to the public information of the entities, and can then decrypt the message, containing the key, which is sent inside, with the secret key he has revealed.
In the protocol 3-KT-3-a(I,R) an adversary proceeds the other way round, that is he reveals the key he needs for the outer decryption and can then get to know the key with the public information he has.
We want to point out that the behavior in the protocols 2-4-b, 2-6-b-4, 2-8-a-1/2, 2-9-b, 2-10, 2-12-a-1/2, and 2-13-b is unexpected in the sense that the claim of secrecy is only violated on one side by this attack.
Attacks after adding SKR: In 2-7(R) and 2-10(R) there is an attack found which runs the whole protocol between another entity and also the entity under test. The test run gets all the messages too, to make him believe he finished the protocol. However the key of the other entity being in the same role as the entity under test is revealed.
In the following attack, all the entities have the same assumption on who is playing what role, but the fact that one party just forwards a message is exploited.

The key can be revealed in the partner's role after he sent the last message and to make the claiming entity believe he properly finished the protocol, the message from the third party is directly sent to the test threat. Similarly, this is observed if only part of the message is forwarded. This attack is observed in 2-8-a/b-1/2(I,R), 2-8-c(R), 2-9-a-1/2/b(I,R), 2-11(R), 2-12-a-1/2(I,R), 2-12-b(R), 2-13-a-1/2(I,R) and 2-13-b(I).
In the protocol 2-13-b(R), there is an attack found making use of the symmetry of the message sent to the third party. This makes it possible to reveal the key from a fake partner which was not intended, who is the third party in another run.
In the next case, the adversary can reveal the keying material in another run where he sends a random nonce as text, because the text field of the message can be changed. This is possible in 2-2(R), 3-KA-6-a/b/c(I,R), 3-KT-2-a/d/v(R), 3-KT-2-c(I,R), 3-KT-3-a(R), 3-KT-4-a/b/c(I,R) and 3-KT-6-a(I,R).
In 2-3-b(R), the claiming entity gets messages from himself in the partner role (from another run), which is possible because of symmetry. The key then gets revealed in this other run. A similar attack is observed in 2-4-b(I,R) and 2-6-b-4(I,R), where the partner does not have to be around in the other protocol.
In 3-KT-6-b, the messages are rerouted over several runs, where in the last step the key is revealed.

6.5 Secrecy given under SKR

All the protocols found to provide secrecy under *SKR*, additionally are at least also secure against the LKR_{others} rule. Therefore no protocols of the standard fall into this category.

6.6 Secrecy given under SKR, LKRothers

The protocols falling in this category provide secrecy of the established key if it is possible for an adversary to reveal the long-term keys, as well as to reveal the session keys. Again, we want to find out what attacks are found if additional capabilities are added.
Attacks after adding $LKR_{aftercorrect}$: Similar attacks are found as when this is added to the group where only LKR_{others} holds. Either the keying material was transported encrypted with the key that then gets revealed (2-3-a(I,R), 2-4-a(I), 2-5-a/b(I), 2-6-a/v(R)) or the outer message can be decrypted with the public information of the entities and then this applies to the inner message (3-KT-2-b(I,R), 3-KT-3-b(I,R)).
Attacks after adding LKR_{actor}: In the protocols 2-3-a(I), 2-4-a(I) and 2-5-b(I) the key is again transported in a message encrypted with the revealed key. In 2-3-a(R) and 2-4-a(R) the adversary can send keying material generated by himself and encrypt it with the key he revealed before from the actor. In 2-5-a(I,R), 2-5-b(R), 2-6-a(I,R) and 2-6-v(I,R) he can do the same, but first needs to learn other information which he can decrypt with the revealed key. In 3-KT-2-b(R) and 3-KT-3-b(R) the secret key of the actor gets revealed such that the adversary has access to the keying material transported encrypted with it.
Attacks after adding *RNR*: In all the protocols, as the keying material is a random number it can be directly revealed with this rule.

6.7 Secrecy given under LKRothers, LKRaftercorrect

For the protocols of this category it is possible to allow an adversary with the capabilities LKR_{others} and $LKR_{aftercorrect}$.
Attacks after adding SKR: similar attacks are found as in the group in Section 6.4. In the protocols 3-KT-5-a/b/c, the same message can be sent to another run with changed text parts, where the key can be revealed. In 2-6-b-3, symmetry is exploited in that one entity is in both roles and in one run the key can be revealed.
Attacks after adding LKR_{actor}: In 3-KT-5-a/b/c(R) the secret key of the receiver gets revealed, and the message where the keying material is transported can be decrypted by the adversary. As the adversary learns the shared key in 2-6-b-3 by the LKR_{actor} rule, he can decrypt messages to learn information and encrypt messages with it to send fake messages.

6.8 Secrecy given under LKRothers, LKRaftercorrect, LKRactor

The protocols 3-KA-5-a/b provide secrecy if the long term key reveal rules LKR_{others}, $LKR_{aftercorrect}$ and LKR_{actor} are enabled. If, however, the SKR rule is added to the adversary capabilities, the session key can be revealed at the end of the session.
If in 3-KA-5-a(I,R) and 3-KA-5-b(R) the RNR rule is added, one of the private key agreement keys can be revealed because they are random numbers, and with this information the session key can be computed. In the attack for 3-KA-5-b(I), the adversary can send his own keying material, because he can use the revealed random number to generate the expected MAC.

6.9 Secrecy given under LKRothers, LKRaftercorrect, SKR

The two variants of the protocol 2-6, 2-6-b-1/2 are secure if LKR_{others}, $LKR_{aftercorrect}$ and SKR are possible. If however, additionally LKR_{actor} is enabled, the adversary is able to send a keying material chosen by him, because he can encrypt it with the shared key he revealed.
Since the keying material is considered a random number, this can be revealed directly if the RNR rule is added.

6.10 Secrecy given under LKRothers, SKR, LKRactor

The protocols 3-KA-3-a/b provide secrecy under the rules LKR_{others}, LKR_{actor} and SKR.
If the RNR rule is added, the adversary can learn the random private key agreement key and compute the key. With the $LKR_{aftercorrect}$ rule added, an adversary can learn the secret key of the initiator and compute the key with this information and the send message.

6.11 Secrecy given under LKRothers SKR RNR

2-1, 3-KA-1 and 3-KA-8 fall in the category which preserves the secrecy of the new key against this three adversary-compromise rules.

If one of the rules $LKR_{aftercorrect}$ or LKR_{actor} is added, the shared long term key is revealed in 2-1 and can be used by an adversary as input to a KDF which will provide him with the new key. If one of these rules is added in 3-KA-1, the key can be computed with the secret key revealed. The same is true if the rule $LKR_{aftercorrect}$ is added in 3-KA-8.
In 3-KA-8(I) revealing random numbers preserves secrecy, however if the LKR_{actor} rule is additionally enabled, an adversary can use the secret key and the random private key agreement key of the initiator to compute the key. In 3-KA-8(R) this rule also reveals the secret key of the receiver, which makes it directly possible to compute the key.
The public key transport protocols which also fall in this category, are treated in the Section 6.15 as a special case.

6.12 Secrecy given under LKRothers LKRafter SKR SSRinfer=2, LKRothers LKRactor LKRaftercorrect SKR SSRinfer=2

The only protocols falling in this category are the two variants 3-KA-7-a/b. *SSR* denotes session state reveal and is not further considered in our analysis. The protocols preserve secrecy under the models LKR_{others}, LKR_{after}, *SKR* and LKR_{others}, LKR_{actor}, $LKR_{aftercorrect}$, *SKR*. If instead of $LKR_{aftercorrect}$, the LKR_{after} rule is combined with the LKR_{actor} rule, secrecy is still preserved. In all these cases, if the *RNR* rule is allowed an adversary can reveal the secretly generated numbers of the entities he needs to compute the key.

6.13 Secrecy given under LKRothers SKR RNR, LKRothers LKRaftercorrect LKRactor SKR

The protocol 3-KA-9 provides secrecy if the adversary is capable of either LKR_{others}, *SKR*, *RNR* or LKR_{others}, $LKR_{aftercorrect}$, LKR_{actor} , *SKR*. If, however the rule *RNR* is enabled together with LKR_{actor} or $LKR_{aftercorrect}$, the adversary can learn both the secret key and the private key agreement key of one of the entities, which enables him to compute the key.
In either case, if additionally the rule LKR_{after} is enabled, the adversary can reveal the session key of a partner who is never present, and construct the sent key token himself.

6.14 Secrecy given under LKRothers SKR RNR, LKRothers LKRaftercorrect LKRactor SKR, LKRafter

In 3-KA-10, additionally to what holds in 3-KA-9 (6.13), allows for the LKR_{after} rule. As in 3-KA-9, if *RNR* is either combined with LKR_{actor}or $LKR_{aftercorrect}$, the adversary can compute the session key by revealing the secret key and the secretly generated number he needs. In the cases where LKR_{after} is enabled at the same time as *RNR*, no attacks are found. Only if LKR_{after} and LKR_{actor} are both enabled at the same time as *RNR*, the adversary can do a similar attack.

6.15 Secrecy in Public Key Transport Protocols

In the Public Key Transport Protocols (presented in Section 4.7) of Part 3 of the standard [10], there is no key established which can be claimed to be secret. As only a public key is sent from one entity to another, we analyzed the secrecy of the corresponding secret key in the protocols 3-PKT-1, 3-PKT-2-a, 3-PKT-2-b and 3-PKT-3. Secrecy is in all cases given under LKR_{others}, as well as when adding the capabilities *SKR* and *RNR*. If however either $LKR_{aftercorrect}$ or LKR_{actor} is added, the secret key gets directly revealed by this rule and secrecy does not hold anymore.

6.16 Conclusion of Hierarchy

Concluding, we observe that a lot of the protocols do not preserve secrecy of the key against an adversary that is capable of more than what was analyzed in Section 5. Still, there are some protocols which make it possible to allow an adversary which also uses the *SKR* rule and $LKR_{aftercorrect}$. In a similar amount of protocols, the rule *RNR* is also possible in addition to *SKR*. For each of the categories discussed in 6.8, 6.9 and 6.10, there is only one protocol, in its two versions, found to preserve secrecy. 3-KA-7 preserves secrecy against LKR_{others}, LKR_{after}, *SKR* and LKR_{others}, LKR_{actor}, $LKR_{aftercorrect}$, *SKR*. Even if the rules LKR_{after} and LKR_{actor} are combined, secrecy is still provided. The protocol 3-KA-9 is found to preserve secrecy against an adversary either capable of LKR_{others}, LKR_{actor}, $LKR_{aftercorrect}$, *SKR* or LKR_{others}, *SKR*, *RNR*. It is not secure, however, if the rule $LKR_{aftercorrect}$ or LKR_{actor} are combined with *RNR*. In 3-KA-10, the same holds but the rule LKR_{after} is additionally possible; if LKR_{after} and LKR_{actor} are enabled at the same time as *RNR* an attack is found.

7 Comparison of Results to Claims of Standard

In this section we want to compare the results to what the standard claims about the protocols. Because there is no defined threat model in the standard with respect to what this comparison can be made, we assume an active adversary that is at least capable of controlling the network. Consequently, we say a claim of the standard is contradicted, if there exists an attack from a Dolev-Yao adversary that contradicts this explicit claim of the standard.

7.1 Confirming the standard

In almost all of the protocols, no attacks are found within bounds for the claim of session key in both roles. The only exceptions are found in 3-KA-2, 3-KA-4, 3-KA-11 and 3-KT-1-a/b. This is only contradicting the explicit statement of the standard in 3-KA-11.
In the protocols 2-1, 2-3-a, 2-3-b, 2-4-a, 2-4-b, 2-5-b, 2-6-a, 2-7, 2-8-c, 2-9-b, 2-13-b, 3-KA-1, 3-KA-2, 3-KA-3-a, 3-KA-4, 3-KA-5-a/b, 3-KA-6-a/b/c, 3-KA-7-a/b, 3-KA-8, 3-KA-9, 3-KA-10, 3-KT-1-a/b and 3-KT-2-a/d no contradiction to any claims explicitly made by the standard have been found. Where authentication is claimed, the protocols fulfill all tested kinds of authentication.

The analysis confirms the standard in that substitution, respectively reflection attacks, are found in 2-3-b, 2-4-b, 2-5-b and 2-6-b-3/4.

In the protocols 2-5-a, 2-8-a-2, 2-9-a-2, 2-12-a-1, 2-12-a-2, 2-13-a-2, 3-KT-2-c, 3-KT-4-c, 3-KT-5-a/b and 3-KT-6-a, the standard is confirmed to some extend in its statement that authentication is given; aliveness and weak agreement are satisfied, but not non-injective agreement and non-injective synchronization.

7.2 Attacks Contradicting the Standard

In the protocols 3-KT-3-a and 3-KT-4-a/b there are attacks found against the claims that entity authentication is achieved, however all claims of key authentication/confirmation are confirmed.

In the protocols 2-9-a-2, 2-12-a-2 and 2-13-a-2 even though non-injective agreement and non-injective synchronization do not hold, the commitment to the same key as the communication partner has is still possible on both sides. In the protocols 2-12-a-1, 3-KA-6-a/b/c, 3-KT-2-a/d,3-KT-3-a/b, 3-KT-4-a/b and 3-KT-5-a/b this is true but only in one entity's claim.

In 3-KA-11 the attack for the session key contradicts the claim that mutual explicit key authentication is given.

Authentication claims of the standard are contradicted in the protocols 2-10, 3-KA-11 and 3-KT-6-a. Fixes to make the protocols achieve the claimed properties are suggested in the Section 8.3.

7.3 Further Attacks

This section treats cases where in one of the variations claimed to be different by the standard, no difference could be seen in the analysis with Scyther.
The standard claims that by adding a MAC in 3-KA-6-c, key confirmation from the responder to the initiator is given. We confirm this, however according to the Scyther analysis, this has already held in the protocol as originally proposed, without the MAC.
In the variations 3-KT-1-b and 3-KA-3-b, the standard says that without the TVP a replay of the key token is possible. This is not confirmed nor contradicted, since all the attacks we found for this version are not a replay of the key token, they are the same as already found in 3-KT-1-a (also see ACP attacks in Section 5.3). The same is true in the variation 3-KT-2-d where the standard says that an additional TVP should be included in the block BE. Again, none of the attacks discovered in the originally proposed protocol are a replay of this block, so all attacks remain the same and no final conclusion can be made.
Also in the hierarchy no difference can be seen between these variants and the respective original protocol.

7.4 Strengthening Claims

This paragraph presents protocols where the standard achieves more than it claims. In the protocols 2-2 and 2-12-b it is claimed that no authentication is given, however at least one entity can claim aliveness and weak agreement of its communication partner. In 3-KT-2-b, it is claimed that authentication is not given from the initiator to the receiver, but no attack has been found. Also, in 3-KT-3-b it is said that no entity authentication is given, which is true only for the initiator while no attacks are found in the responder's claims of authentication.

8 Recommendations

In this section, we present our recommendations based on the above results. All but the last subsection refer to the findings of Section 5, which analyzed the protocols with respect to a Dolev-Yao adversary. The last part provides recommendations concerning other compromising-adversary rules (from Section 6). First, we make recommendations how to detail the specifications in the standard, then we suggest fixes to the protocols that do not satisfy the claims made in the standard. Then, we also recommend improvements to other protocols. Where we propose to change the protocol, the corrected version can be found in [1] denoted by the name of the protocol, followed by "-corr".

8.1 Adding more Security Property Specifications

For the protocols 2-1, 2-2, 2-7 and 2-11 no positive claims are made by the standard. We recommend to add to the description the properties that should be expected. In all these protocols one can make the positive statement that the established key is secret. In 2-2, additionally aliveness and weak agreement from the initiator to the receiver are given and can be added.
In the variations of the protocols there are often no explicit positive statements made; it is only said what does not hold anymore if one omits or adds certain parts. We recommend to explicitly declare that except for the restriction the same claims as originally made still hold or if this is not true to explicitly say what is achieved. This applies in the protocols 2-3-b, 2-4-b, 2-5-b, 2-6-b, 2-6-v, 2-8-c, 2-9-b, 2-12-b, 2-13-b, 3-KT-1-b, 3-KT-2-b, 3-KT-3-b, 3-KT-4-b, 3-KT-5-b and 3-KT-6-b.

In the following protocols the standard claims to achieve authentication and the analysis confirms that aliveness and weak agreement are given, but not stronger forms of authentications. It depends on the use of the protocol whether or not this is enough, but we suggest to add to the description the degree of authentication that can be expected. In other words, we recommend to say in the protocols 2-5-a, 2-8-a-2, 2-9-a-2, 2-12-a-1, 2-12-a-2, 2-13-a-2, 3-KT-2-c, 3-KT-4-c, 3-KT-5-a/b and 3-KT-6-a that aliveness and weak agreement are given, but not stronger forms of authentication.

8.2 Recommended Input to Key Derivation Function

In almost all the protocols which reuse the newly established key, a difference could be observed in whether only the keying material or also the distinguishing identifiers are added to the KDF. In the protocols 2-8-a, 2-9-a and 2-13-a, when using only the keying material as input, in at least one entity's claim even the weakest form of authentication has not held. In all the corresponding versions which additionally input the entity identifiers, there were only attacks found which violate non-injective agreement and non-injective synchronization and except for 2-13-a, the commitment to key and entity. Therefore, we recommend to use a key derivation function which takes the identifiers of the initiator and the responder as additional inputs. In 2-12-a and 2-8-b, where also both these versions were analyzed, the variation taking all three inputs disables an attack against the commitment of one entity to the key/the identity of the partner.

8.3 Proposed Fixes to Protocols

This paragraph suggests fixes to the protocols that contradict the positive claims of the standard.

In the protocols 3-KT-6 and 2-10, the attacks make use of the fact that the direction of the message can not be seen in protocols with three parties. This gives rise to role-mixup attacks. As in the second principle of [4], we suggest to include the identities and the roles to avoid this problem. If in 3-KT-6-a the responder's identity is added, no attacks are found anymore for the claim of the responder to commit to the key. However, the replacement of the unprotected text parts still makes an attack possible in the initiator's claim to commit to the key. If the text fields are not needed, we suggest to omit them. In 3-KT-6-b-corr, where the same fix was added, no attacks are found for all authentication claims. This means that key confirmation is now given as claimed by the standard, as well as a stronger entity authentication that is also claimed. The same applies in 2-10-corr, where in each message the sender's and the receiver's identity is added. This results in a protocol that achieves aliveness, weak agreement and commitment to the partner's identity in all three cases that are claimed by the standard but have been falsified before.
In the protocol 3-KA-11 it is claimed that explicit key authentication is given, but there is an attack found that contradicts secrecy of the key. To provide secrecy of the key also in the responder's role, we suggest to send the first message over an authenticated channel. We additionally recommend to add again the sender's and receiver's identity to the messages. The result is that not only secrecy holds, but all claims of authentication in the initiator's claims, as well as the claims of aliveness, weak agreement and commitment to the key in the responder's role. Since both entities can commit to the key and know that it is secret, this now provides the claimed explicit key authentication.

8.4 Proposed Improvements to Protocols

In 3-KT-1, it is explicitly said by the standard that [10] *"as entity B receives the key K from the non-authenticated entity A, secure usage of K by entity B is restricted to functions not requiring trust in entity A's authenticity"*. Alternatively, to make sure the key is secure, the message 1 can be sent over an authenticated channel.
In the protocols 2-5-a/b we suggest to bind two messages to each other, by including to the second message the *TVP* sent in the first message. The two messages have had the same form before, which is not true anymore and hence they can not be reflected. Additionally, it binds the answer to one specific sent

message, which has the effect that all kinds of authentication are given in the corrected version.
If it is not needed to have a mechanism that allows an entity to communicate with itself, we suggest to avoid this in 2-8-a-2, since then the only attack found is sending a wrong message which the initiator can not encrypt and this only violates non-injective agreement, non-injective synchronization, but not the commitment to the key.
Again applying the second principle of [4], we suggest to also add the sender and receiver in the protocol 3-KT-2-c. Additionally, the TVP sent by the sender should be included in the reply to bind them together. This achieves that the commitment to the other entity's identity as well as to the key is possible. The only attack found is then violating non-injective agreement and non-injective synchronization and is replacing the unprotected text parts.

8.5 Recommendations from the Hierarchy

Since it can be seen in the hierarchy under what adversary model the protocols are secure, we suggest to choose accordingly. If one has a defined adversary model and a subset of its properties match a set of the rules we considered, we recommend to choose a protocol being secure in this adversary model according to the hierarchy in Section 6. If for example one wants to consider an adversary that is able of the rules LKR_{others}, $LKR_{aftercorrect}$and SKR, we suggest to take either of the protocols 2-6-b-1/2 (also see Section 6.9).

Since the protocols 3-KA-2, 3-KA-4, 3-KA-11 and 3-KT-1 do not preserve secrecy under any model, we recommend to not use them if this property is of concern. If one considers a Dolev-Yao adversary, any other protocol preserves secrecy.
If it is required that the protocols are secure against an adversary that is also capable of SKR or $LKR_{aftercorrect}$, we suggest to chose a protocol from the Section 6.6 or 6.7 respectively.
In the protocols presented in Section 6.6 where the rules LKR_{others} and SKR are possible, if the cost of additional messages is of concern, we suggest to take one of the protocols 2-3-a, 3-KT-2-b or 3-KT-3-b, only sending one message. If however it is more of concern that strong forms of authentication are given and it is not expensive to have shared keys distributed in advance, we suggest to use 2-6-a.
Within the category presented in Section 6.7 where LKR_{others} and $LKR_{aftercorrect}$ are enabled, all protocols have the same amount of messages exchanged. 3-KT-5-a/b, however, gives stronger authentication guarantees and has the additional advantage that no shared keys need to be pre-distributed.
Both protocols allowing an adversary being capable of LKR_{others}, SKR and RNR (the public key transport protocols excluded), do not provide entity authentication. Also both protocols, require the pre-distribution of either a shared key or public key agreement keys. Since 3-KA-1 does not require a message exchange, we recommend to choose this one if communication is expensive.
If the considered adversary matches a subset of the rules presented in 6.8, 6.9, 6.10, 6.12, 6.13, or 6.14, we recommend to choose a protocol falling in this category. As in these categories there is only one protocol, sometimes in two versions, no further preference with respect to amount of messages or pre-distributed keys

can be made.

9 Comparison to Previous Work

This section provides the reader with a comparison to the work of Tomas Zgraggen [12] where he analyzed the same protocols in his Bachelor thesis. Instead of Scyther, he was using the automated proof tool Tamarin to analyze the security properties of the protocols. In the following comparison, we chose to keep the nomenclature we introduced even though it is not matching one by one with the one used in the old work.

9.1 Difference in set of modeled protocols

For the following protocols no comparison was possible, because they have not been analyzed in the previous work. This is true for the protocols 3-KA-8, 3-KA-9 and 3-KA-10. The same holds for the variations of the protocols 2-6-b-1/2, 2-6-v, 2-8-b, 3-KA-6-b/c, 3-KT-2-c/d/v, 3-KT-4-b/c and 3-KT-5-b/c.

9.2 Additional Positive Findings

The analysis by Scyther confirms the statement that in 2-1 no authentication claims hold. In the old work no explicit statement about secrecy is made. However, the findings of Scyther imply that the statement of [12] which says *"no security guarantees can be made"* is not correct, since secrecy of the key is given.

9.3 Similar Findings

We confirm the previous work in that there are no attacks found for both variants of the protocol 3-KA-7.
In the protocols 2-3-a, 2-6-b-3/4 and 2-10 the description of the attacks in the old thesis matches the attacks discovered by Scyther.
We also confirm the previous work in that there is an attack found in 3-KA-1, which is already mentioned in the standard.
We confirm the findings in protocols 2-3-b, 2-4-b and 2-5-b that the substitution attacks mentioned in the standard are found. Nevertheless, there is a disagreement on the statement that this also constitutes an unknown key sharing attack. The definition of [12] of an unknown key sharing attack is *"where either an actor C believes to share a key with A, when that key is actually shared between A and B, or A believes she shares a key with B, while B has never participated in the protocol."* We exclude the second case in our definition. Therefore, we do not consider these attacks to also constitute a unknown key sharing attacks. As it is claimed by the standard that these protocols are only secure in an environment where the substitution attacks are not possible, we draw the additional conclusion that in these environments also the unknown key sharing attacks as defined by [12] are disabled.
In 2-7, we confirm the RMU attacks found, and we also find the UKS attack mentioned, which again is not considered an UKS but rather a ACP attack in our definitions (explained in Section 5.3). Additionally, in our case the attack

described as UKS in the old work does not require the KDC to be able to act as a normal entity. The same applies for the UKS that has been found in the old work for 2-9-b.
In 2-8-c and 2-11, the old description of the attack matches what is found in the responder's claims in our analysis. Additionally, we found attacks violating the initiator's claims that do not require the third party to be able to act as a normal party.
For the attacks of 3-KT-1-a, we confirm that no authentication is given, and we additionally found an attack violating the session key claim.
In 3-KA-2 in the previous work, there is a *"key authentication attack"* found to violate the standard's claim that implicit key authentication is given from the receiver to the sender. We find the described attack but in the receiver's claim of session key instead. This means, unlike the previous work, we confirm the standard in that implicit key authentication is given from the receiver to the initiator, since no attack is found in the initiator's claim of session key. We believe that the described attack violates the claim of implicit key authentication in the other direction, which was not claimed by the standard.

9.4 New Attacks found

The protocols 2-4-a, 2-5-a, 3-KA-6-a, 3-KA-11 and 3-KT-5-a were verified in the old work in the regular model but we found new attacks in all of them. In 3-KT-5-a, the only attack we found is to change the unprotected text parts. In 2-4-a, however, we found ACP attacks (Section 5.3) violating even the weakest claims of authentication. In 2-5-a and 3-KA-6-a we found 1En2Ro (Section 5.6), respectively RepText (Section 5.4) and ACP attacks (Section 5.3) not mentioned in the old work. In 3-KA-11, we not only found attacks for all the authentication claims except for aliveness, but also for the claim of secrecy (Section 5.10.

The protocols 2-6-a, 2-9-a, 2-12-a/b, 2-13-a/b, 3-KA-3-b, 3-KA-5, 3-KT-2-a/b, 3-KT-3-a/b, 3-KT-4-a and 3-KT-6-a/b did not terminate in the previous work, so no attacks have been found before. Except for 2-6-a, we find new attacks in all of them (see tables 3-8).

9.5 Attacks not considered in this work

In the protocols 2-10 and 2-11, there was a type flaw attack found in the old work, which we did not explore.

10 Future Work

Because of time constraints, we only considered secrecy with respect to the different compromise adversary rules, as presented in Section 6. We leave it to future work to examine what other security properties, such as entity and key authentication, still hold under what adversary assumptions. It could then also be analyzed with respect to what adversary the claims of the standard still hold.

In almost all key agreement protocols, a lot of approximations were needed to model the arithmetic. It would be interesting to see if further attacks are found with yet another tool that has better support for these calculations.

11 Conclusion

Examining the protocols of Part 2 and 3 of the ISO/IEC 11770 in a Dolev-Yao adversary model, we have found that some of the protocols meet the claimed specifications, while others only fulfill them to some degree or contradict the explicit claims of the standard. In the cases where the attacks of an active adversary violate what is explicitly claimed by the standard, we have provided fixes. We have also made other improvement suggestions; often we have recommended to clarify the specifications for making it clear against what adversary model the protocol achieves which security goals.
Additionally, we have provided a hierarchy to illustrate what model preserves secrecy under what adversary-compromise rules.

Based on our analysis, we recommend avoiding for now the protocols 2-10 (Key Establishment Mechanism 10 of [9]), 3-KA-11 (Key Agreement Mechanism 11 in [10]) and 3-KT-6-a (Key Transport Mechanism 6 in [10]) as they appear in the standard, because they do not meet the claimed requirements with respect to an adversary that controls the network. For details of the attacks found, we refer to Section 5.4, 5.8, 5.10 and 7.2; for proposed new versions of the protocols that meet at least some degree of the claimed properties, we refer to Section 8.3.
We suggest to only choose the protocols 2-5-a, 2-8-a-2, 2-9-a-2, 2-12-a-1, 2-12-a-2 and 2-13-a-2 (Key Establishment Mechanisms 5, 8, 9, 12 and 13 in [9]), as well as 3-KT-2-c, 3-KT-4-c, 3-KT-5-a/b and 3-KT-6-a (Key Transport Mechanisms 2, 4, 5 and 6 in [10]) if aliveness and weak agreement are sufficient properties for the designated purpose, as these protocols do not fulfill a higher degree of the claimed entity authentication in this variation (also see Section 8.1, and Section 2.3 for the protocol variations).
Further, we advise that care should be taken in using protocols with no positive claims on what is achieved and refer to Section 8.1 to find a possible description of what can be expected from these protocols.
If one has a defined adversary model and a subset of the properties can be found in our analysis of Section 6, we suggest to choose a protocol that preserves secrecy according to the hierarchy provided. For a summary of recommendations with respect to this hierarchy, we refer to Section 8.5.

We conclude that care should be taken at consulting the standard, not only because in some cases claims do not hold under the assumption that an adversary controls the network, but also because missing specifications can lead to wrong conclusions what security properties are provided.

References

[1] Repository of models of the iso/iec-11770 mechanisms. `https://svn.inf.ethz.ch/svn/basin/infsec/trunk/teaching/theses/bachelorarbeiten/2013_iso11770/thesis/Models/finalmodels/`, retrieved in June 2013.

[2] David Basin and Cas Cremers. Modeling and analyzing security in the presence of compromising adversaries.

[3] David Basin and Cas Cremers. Degrees of security: protocol guarantees in the face of compromising adversaries. In *Proceedings of the 24th international conference/19th annual conference on Computer science logic*, CSL'10/EACSL'10, pages 1–18, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] David Basin, Cas Cremers, and Simon Meier. Provably repairing the iso/iec 9798 standard for entity authentication. In *Proceedings of the First international conference on Principles of Security and Trust*, POST'12, pages 129–148, Berlin, Heidelberg, 2012. Springer-Verlag.

[5] Cas Cremers. Scyther tool. `http://people.inf.ethz.ch/cremersc/scyther/index.html`.

[6] Cas Cremers and Sjouke Mauw. *Operational Semantics and Verification of Security Protocols.* Information Security and Cryptography. Springer, 2012.

[7] Cas JF Cremers. The scyther tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification*, pages 414–418. Springer, 2008.

[8] Gavin Lowe. A hierarchy of authentication specifications. pages 31–43. IEEE Computer Society Press, 1997.

[9] International Organization of Standardization. Iso/iec 11770, information technology - security techniques - key management - part 2 : Mechanisms using symmetric techniques. 2008.

[10] International Organization of Standardization. Iso/iec 11770, information technology - security techniques - key management - part 3 : Mechanisms using asymmetric techniques. 2008.

[11] International Organization of Standardization. Iso/iec 11770, information technology - security techniques - key management - part 1 : Framework. revision of 2010.

[12] Tomas Zgraggen. Analysing and repairing the iso 11770 standard for key management. Bachelor Thesis ETH Zurich, July 2012.

Protocol 2-4-a with the optional identifier

```
hashfunction KDF;
usertype KeyingMaterial;
protocol isoiec-11770-2-4-a(A,B)
{
        role A
        {        //declaration of variables and fresh values
                var Rb: Nonce;
                fresh Text1: Ticket;
                fresh F: KeyingMaterial;

                //message exchange from the view of the initiator
                recv_1(B, A, Rb);
                claim(A, Running, B, A);
                claim(A,Running,B,KDF(F));
                send_2(A, B, {Rb, B, F, Text1}k(A,B));

                // claims made by entity in role A
                claim(A, SKR, KDF(F));
                claim(A, Alive);
                claim(A,Weakagree);
                claim(A, Niagree);
                claim(A, Nisynch);
                claim(A,Commit,B,B);
        }
        role B
        {        fresh Rb: Nonce;
                var Text1: Ticket;
                var F: KeyingMaterial;

                claim(B,Running,A,B);
                send_1(B, A, Rb);
                recv_2(A, B, {Rb, B, F, Text1}k(A,B));

                claim(B, SKR, KDF(F));
                claim(B,Commit,A,KDF(F));
                claim(B, Alive);
                claim(B,Weakagree);
                claim(B, Niagree);
                claim(B, Nisynch);
                claim(B,Commit,A,A);
        }
}
//helper protocol to make the key symmetric
protocol @keysymm-2-4-a(A,B)
{
        role A
        {        var R: Nonce;
                var Text: Ticket;
                var F: KeyingMaterial;
                recv_!1(B,A, { R, A, F, Text }k(A,B) );
                send_!2(A,B, { R, A, F, Text }k(B,A) );
        }
        role B
        {        var R: Nonce;
                var Text: Ticket;
                var F: KeyingMaterial;
                recv_!3(A,B, { R, B, F, Text }k(A,B) );
                send_!4(B,A, { R, B, F, Text }k(B,A) );
        }
}
```

Figure 7: Example of input provided to the Scyther tool.

```
                                 Model of the Protocol 2-10
hashfunction KDF;
usertype KeyingMaterial;
protocol isoiec-11770-2-10(A,B,P)
{
        role A
        {        fresh TNA: Nonce;
                fresh Text1: Ticket;
                var TNP: Nonce;
                var F: KeyingMaterial;
                var Text2: Ticket;

                claim(A, Running, P, A);
                claim(A,Running,B,A);
                send_1(A, P, {TNA, B, Text1}k(A,P));
                recv_2(P, A, {TNP, F, B, Text2}k(A,P));

                claim(A, SKR, KDF(F));
                claim(A, Alive, P);
                claim(A, Weakagree, P);
                claim(A, Alive, B);
                claim(A, Alive);
                claim(A, Weakagree);
                claim(A, Nisynch);
                claim(A, Niagree);
                claim(A, Commit, P, P);
        }
        role B
        {        var TNP2: Nonce;
                var F: KeyingMaterial;
                var Text3: Ticket;

                recv_3(P, B, {TNP2, F, A, Text3}k(B,P));

                claim(B,  SKR, KDF(F));
                claim(B, Alive, P);
                claim(B, Weakagree, P);
                claim(B, Alive);
                claim(B, Weakagree);
                claim(B, Nisynch);
                claim(B, Niagree);
                claim(B, Commit, P,P);
                claim(B,Alive,A);
                claim(B,Weakagree,A);
                claim(B,Commit,A,A);
        }
        role P
        {
                var TNA: Nonce;
                var Text1: Ticket;
                fresh F: KeyingMaterial;
                fresh TNP: Nonce;
                fresh Text2: Ticket;
                fresh TNP2: Nonce;
                fresh Text3: Ticket;

                recv_1(A, P, {TNA, B, Text1}k(A,P));
                claim(P, Running, A, P);
                send_2(P, A, {TNP, F, B, Text2}k(A,P));
                claim(P,Running,B,P);
                send_3(P, B, {TNP2, F, A, Text3}k(B,P));

                claim(P, Alive,A);
                claim(P, Weakagree,A);
                claim(P, Alive);
                claim(P, Weakagree);
                claim(P, Commit, A, A);

        }
}
```

Figure 8: Model 2-10 input to the Scyther tool.

Model of the Protocol 3-KA-11

```
hashfunction KDF;
hashfunction MAC;

macro M1= rA,Text1;
macro CertB = pk(B);
macro M2 = rB, CertB, Text2;
macro KTA2= {rA2}pk(B);

protocol isoiec-11770-3-KA-11(A,B)
{
        role A
        {        fresh rA,rA2:Nonce;
                fresh Text1:Ticket;
                var rB:Nonce;
                var Text2:Ticket;

                send_1(A,B, M1);
                recv_2(B,A, M2);
                claim(A,Running, B, KDF( rA,rB,rA2));
                send_3(A,B, KTA2,MAC(M1,KTA2,KDF( rA,rB,rA2)));
                recv_4(B,A, MAC(M2,KDF( rA,rB,rA2)));

                claim(A, Secret, KDF( rA,rB,rA2));
                claim(A, SKR, KDF( rA,rB,rA2));
                claim(A, Commit, B, KDF( rA,rB,rA2));
                claim(A,Alive);
                claim(A,Weakagree);
                claim(A,Niagree);
                claim(A,Nisynch);

        }
        role B
        {        var rA,rA2:Nonce;
                var Text1:Ticket;
                fresh rB:Nonce;
                fresh Text2:Ticket;

                recv_1(A,B, M1);
                send_2(B,A, M2);
                recv_3(A,B, KTA2,MAC(M1,KTA2,KDF( rA,rB,rA2)));
                claim(B,Running,A,KDF( rA,rB,rA2));
                send_4(B,A, MAC(M2,KDF( rA,rB,rA2)));

                claim(B, Secret, KDF( rA,rB,rA2));
                claim(B,SKR, KDF(rA,rB,rA2));
                claim(B, Commit, A, KDF( rA,rB,rA2));
                claim(B,Alive);
                claim(B,Weakagree);
                claim(B,Niagree);
                claim(B,Nisynch);
        }
}
```

Figure 9: Model 3-KA-11 input to the Scyther tool.