INTERNATIONAL STANDARD

ISO/IEC 29128

First edition
2011-12-15

Information technology — Security techniques — Verification of cryptographic protocols

Technologies de l'information — Techniques de sécurité — Vérification des protocoles cryptographiques

Reference number
ISO/IEC 29128:2011(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 29128 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

Introduction

The security of digital communications is dependent on a number of aspects, where cryptographic mechanisms play an increasingly important role. When such mechanisms are being used, there are a number of security concerns such as the strength of the cryptographic algorithms, the accuracy and correctness of the implementation, the correct operation and use of cryptographic systems, and the security of the deployed cryptographic protocols.

Standards already exist for the specification of cryptographic algorithms, and for the implementation and test of cryptographic devices and modules. However, there are no standards or generally accepted processes for the assessment of the specification of protocols used in such communication. The goal of this International Standard is to establish means for verification of cryptographic protocol specifications to provide defined levels of confidence concerning the security of the specification of cryptographic protocols.

Information technology — Security techniques — Verification of cryptographic protocols

1 Scope

This International Standard establishes a technical base for the security proof of the specification of cryptographic protocols. This International Standard specifies design evaluation criteria for these protocols, as well as methods to be applied in a verification process for such protocols. This International Standard also provides definitions of different protocol assurance levels consistent with evaluation assurance components in ISO/IEC 15408.

2 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

2.1

arity

number of arguments

2.2

cryptographic protocol

protocol which performs a security-related function using cryptography

2.3

formal methods

techniques based on well-established mathematical concepts for modelling, calculation, and predication used in the specification, design, analysis, construction, and assurance of hardware and software systems

2.4

formal description

description whose syntax and semantics are defined on the basis of well-established mathematical concepts

2.5

formal language

language for modelling, calculation, and predication in the specification, design, analysis, construction, and assurance of hardware and software systems whose syntax and semantics are defined on the basis of well-established mathematical concepts

2.6

adversarial model

description of the powers of adversaries who can try to defeat the protocol

NOTE It includes restriction on available resources, ability of adversaries, etc.

2.7

security property

formally or informally defined property which a cryptographic protocol is designed to assure such as secrecy, authenticity, or anonymity

2.8

self-assessment evidence

evidence that the developer uses to verify whether a specified protocol fulfils its designated security properties

NOTE It includes cryptographic protocol specification, adversarial model and output (transcripts) of formal verification tool.

2.9

protocol model

specification of a protocol and its behaviour with respect to an adversarial model

2.10

protocol specification

all formal and informal descriptions of a specified protocol

NOTE It includes all processes by each protocol participant, all communications between them and their order.

2.11

secrecy

security property for a cryptographic protocol stating that a message or data should not be learned by unauthorized entities

2.12

variadic

property of a function whose arity is variable

3 Symbols and notation

For the purposes of this document, the following symbols and notation apply.

ϕ security property of a protocol model

A,B	role names
m	message
r	random nonce
k	key
c	communication channel
enc	encryption function
dec	decryption function
<..,…>	paring operator
Send	sending process
Receive	receiving process

4 General

Verification of a cryptographic protocol involves checking the following artifacts:

a) specification of the cryptographic protocol;

b) specification of the adversarial model;

c) specification of the security objectives and properties;

d) self-assessment evidence that the specification of the cryptographic protocol in its adversarial model achieves and satisfies its objectives and properties.

The artifacts shall clearly state parameters or properties relevant for the verification. Examples include the bound used in bounded verification as later descibed in Clause 7.4.4.1 or assumed algebraic properties of cryptographic operators used in the protocol as described in Clause 7.1.2.3 and Clause 5.3.4.

The different Protocol Assurance Levels will lead to different requirements for these four artifacts. The stated requirements are only for design verification not implementation verification.

NOTE 1 For verifying an implementation, additional assurance requirements should be supplied and satisfied.

This International Standard does not specify precisely what proof methods or tools shall be used, but instead only specifes their properties. This encourages protocol designers to use the state-of-the-art for protocol verification in terms of models, methods, and tools.

Verification tools shall fulfil the following conditions.

a) The verification tools are sound.

The protocol designer or possibly an independent third party shall provide evidence of the correctness of the verification tool used. This may, for example, be in terms of a pencil-and-paper proof of the soundness of the calculus used or, in some cases, in terms of code inspection to see that the tool properly implements the calculus.

NOTE 2 This step is nontrivial, yet it is essential if machine checked proofs are to provide greater confidence than hand proofs. In theory, this can be done once and for all for a verification tool, although in practice, tools evolve over time.

b) The results of verification tools are documented in such a way that they are repeatable.

The protocol designer shall provide adequate documentation, including all inputs needed for the tool to construct a proof or (in the case of decision procedures) determine provability.

c) The verification tools are available for outside evaluation and use.

The protocol designer shall indicate all necessary tools to independently check the proofs.

NOTE 3 At least in theory, protocol verification canbe carried out by hand proofs, using paper and pencil. However, given the substantial amount of detail typically involved in security protocol verification, especially for the high Protocol Assurance Levels, confidence in the results is substantially increased by using mechanized tools such as model checkers and theorem provers. Thus, proofs only with paper-and-pencil are treated as lower assurance level (i.e. PAL2) than mechanized proof in this International Standard.

5 Specifying cryptographic protocols

5.1 Objectives

The goal of this part is to provide guidelines and minimal requirements for specifying cryptographic protocols.

5.2 The abstraction levels

The protocols can be specified at several levels of abstraction, each corresponding to a computation model. At the most abstract level, messages are terms constructed from symbols and the attacker is also modelled as a formal process. We will call this abstraction the *symbolic* level. In such a model, the resources (both time and space resources) are not considered.

Any other model can be defined as a refinement of a symbolic model. For instance we can interpret the symbols used in the symbolic model as functions on bitstrings, that can be computed in polynomial time.

Therefore, any cryptographic protocol consists in a symbolic specification and an interpretation in a given domain (e.g. bitstrings or structured data, or even material-dependent formats) of all the symbols, together with assumptions on their interpretation. Such hypotheses can ensure some correspondence between the properties at various abstraction levels.

NOTE In this International Standard, we only consider the symbolic specification of security protocols.

Further documents are required for the specification of other (lower) abstraction levels. Typically, it will be necessary to explain how to specify the interpretation domain and how to carry security guarantees across levels of abstraction.

5.3 The specification of security protocols

5.3.1 General

As explained, a symbolic specification is the necessary first part towards the full specification of a protocol. We list below the minimal mandatory parts in a symbolic protocol specification.

5.3.2 The symbolic messages

The first part consists in specifying what are the possible (valid) messages.

In this clause, the cryptographic primitives used in the protocol must be listed. Since we are talking about a symbolic specification, this part consists of providing with

1. a set of *function symbols* $\mathcal{F}$

   Each function symbol has either a fixed *arity*, that has to be specified, or it is variadic (in which case it has also to be specified).

2. a set of *name symbols* $\mathcal{N}$ that may be split into various syntactic categories that have to be specified.

3. a set of *variable symbols* $\mathcal{X}$.

4. a formal description of valid rules allowing to build messages using the function symbols.
   A (non exhaustive) list of possible ways to specify such a language is:

— Nothing: all terms that are built with the function symbols and following the arity restrictions are valid messages

— Some arguments are restricted to names: some of the arguments of function symbols are restricted to belong to some name categories

— Sorts: a type discipline is defined and only well-typed terms correspond to messages.

The set of valid terms (or messages) is written $(\mathcal{F}, \mathcal{N})$ (or $(\mathcal{F}, \mathcal{N}, \mathcal{X})$ when variables are involved)

EXAMPLE A typical example is encryption, that can be modeled by a symbol `enc`, whose arity is 2 or 3 (or 4), depending on whether the random seed is explicit or not (and whether the encryption algorithm is explicit or not). A specification has to make precise what is the arity of `enc` and what are the assumed types of its arguments. Typically, `enc` has an arity 3. As possible name categories there are the random seeds, whose symbols will start with r, the keys, whose symbols will start with k, the algorithms, whose symbols will start with α, and so on. If `enc` has been specified as being an arity 3 symbol, we can additionally restrict its arguments, specifying for instance that the first argument must be a key and the last one must be a random. In that case, $\mathtt{enc}(k, k, r)$, $\mathtt{enc}(k, \mathtt{enc}(k, r_1, r_2), r_1)$, are valid terms while $\mathtt{enc}(\mathtt{enc}(k, k', r), k'', r')$ and $\mathtt{enc}(r, k, r')$ are not valid terms. Examples of symbols that can be considered as variadicinclude the exclusive or $\oplus$, the arithmetic multiplication × or the concatenation $\|$.

5.3.3 Observing messages

This part consists of specifying some comparison predicates between messages.

Only the equality predicate is mandatory, since other predicates could be seen as Boolean functions and specified within the equality definition. It might however be useful to distinguish later between the computation abilities and the observation abilities. Moreover, in many current specification languages, properties of the function symbols are specified equationally (see clause 5.3.4), while it might be impossible to specify equationally the observation abilities.

This part consists in listing predicate symbols, together with their arity. Typical examples include typing predicates, equality, and `same_length` (that checks that its two arguments have the same length), `same_key` (that checks whether two ciphertexts are encrypted with the same key).

5.3.4 Algebraic properties

This part specifies when two valid terms represent the same message and, more generally, what are the interpretations of the predicate symbols listed in the previous clause.

For instance, when function symbols include both (symmetric) encryption and decryption, we might wish to state that $\mathtt{dec}(k, \mathtt{enc}(k, x, r)) = x$ where r is a symbol for a random seed to express probabilistic encryption: these are two term representations of the same message. We might also wish to state that, $x \oplus x = 0$ if we are using a symbol $\oplus$ meant to represent exclusive or.

As usual, we assume that any two terms that are not specified to be equal are different. The same rule applies to the predicates: everything that has not been specified to be true is, by default, false.

5.3.5 Protocol roles

A *role* is an interactive program that receives some input from the environment and sends messages to the environment. This is the atomic program component of a protocol: there is no communication that takes place inside a role.

Specifying a role requires to provide with:

1. A role name;

2. A finite list of formal parameters: these are the data, that can be used by the program without being generated or received from the environment;

3. A (usually finite, but this is not mandatory) set of control states;

4. A finite set of local variables and local names;

5. A specification of the sending and receiving abilities, as well as state transitions;

6. Formally, this amounts to specify two relations $q,v \xrightarrow{\mathrm{Send}(c,m)} q',v'$ and $q,v \xrightarrow{\mathrm{Receive}(c,m)} q',v'$, a communication channel c and a message m.

Such a specification does not commit to any particular programming language or any particular way to perform tests or moves. It only requires the specification of import/export data and communications with the environment.

EXAMPLE This is a possible specification of the responder role in the public key Needham-Schroeder protocol. We assume a single communication channel, which is omitted below.

1. role name: B;

2. parameters: the identity b of the agent running the instance of this role, the identity a, the private key of b, the public key of a;

3. local states: there are only three local states: q_0, q_1, q_f ;

4. local variables and names: n_B, r are local names and x, y are local variables

A specification of the transitions. Any other formally defined language can be substituted here:

$$q_0, n_B, r, x = 0, \mathrm{y} = 0 \xrightarrow{\mathtt{Receive}(m)} q_1, n_B, r, x = m, y = 0$$

$$q_1, n_B, r, x = m, y = 0 \xrightarrow{\mathtt{Send}\left(\mathbf{enc}\left(\mathbf{pub}(a), \langle \mathrm{m}', \mathrm{n} \rangle, \mathrm{r}\right)\right)} q_f, n_B, x = m, y = m'$$

$$\text{if } a = \pi_1\left(\mathbf{dec}(\mathtt{priv}(\mathrm{b}), x)\right), m' = \pi_2(\mathbf{dec}(\mathtt{priv}(b), x)\,)$$

We use here a ternary encryption symbol `enc`, a decryption symbol `dec`, a pairing symbol ⟨_,_⟩ and projections symbols π_1, π_2.

NOTE In such an example, the transition is not specified when the test fails, meaning that there is no transition in this case: the program is stacked in state q_1. There are of course many other possible designs.

Sessions A *role instance* is a specific copy of a role, together with its actual parameters. This is sometimes called a *session*. As the same identity can run concurrently several copies of the same role, it might be convenient to include in the parameters an *identifier* also called sometimes *session number*, that will allow different role instances to be distinguished.

5.4 The specification of adversarial model

5.4.1 Network specification

This part specifies what are the (symbolic) communications devices and their reliability.

Typically, a list of channels (terms or symbols) is given, each of which with its own properties. For instance, one could specify a single public communication channel c which is under complete control of the attacker (who can intercept messages and send fake messages). But it could be refined further, distinguishing a more (or less!) secure proxy communication channel or a wireless channel that can only be eavesdropped, or even a private channel that is completely out of control of the attacker.

5.4.2 The attacker

This part specifies the computational abilities of the symbolic attacker. In other words, it specifies the messages m that can be computed from a set of messages S.

Typical specifications use inference systems such as the “Dolev-Yao inference system”. These rules might depend on whether there is an explicit decryption symbol or not, for instance. The simplest specification consists in having all function symbols explicit and public. Then the attacker, when given a set of names $\mathcal{N}$, is able to compute any term built on $\boldsymbol{\mathcal{F}}$ and $\mathcal{N}$.

In addition, the attacker can use the predicate symbols of clause 5.3.3, even though such predicates are not used in the definition of the roles. Attacker's other capabilities are specified in clause 5.4.1 and depend on the reliability of the various channels.

From this clause and the previous one, it should be possible to define formally what are the possible execution traces.

NOTE 1 Typical attacks can be formally described as follows.

Eavesdropping attack is a typical security risk posed to networks. In some network environment, messages are broadcasted to everyone. This often can cause a problem that important messages such as passwords and credit card numbers might be delivered to unintended person. This attack can be formally described as a subset of the model in clause 5.4.1. That is, given a list of channels $\{c\}$, an attacker has no control on the channels $\{c\}$ but can listen to all messages S exchanged over the channels. Then, the whole knowledge of the attacker is any term which can be computed from a set of messages S.

Replay attack is another type of risk. In open networks like Internet, messages can be exchanged via routers which is under control of malicious person. In such an environment, messages such as passwords or credit card numbers exchanged over a network might be maliciously repeated or delayed by them. This often can cause a problem that unintended person impersonates a legitimate one by repeating the stored password as a proof of identity. This attack can be formally described as the model in clause 5.4.1 and a subset of the model in clause 5.4.1. That is, given a list of channels $\{c\}$ and a set of messages S exchanged over the channels $\{c\}$, an attacker has complete control on the channels $\{c\}$ and listen to all messages S. Thus, the whole knowledge of the attacker is any term which can be computed from a set of messages S. But, in replay attack, he uses only the elements of S and tries to impersonate some role.

NOTE 2 Extension of the model is required to describe a series of attacks such as denial-of-service attack and relay attack. Since these attacks are related to the physical properties of an actual system such as processing time of operations and communication speed via physical media, in order to describe such attacks, such physical properties should be somehow included in the extended model.

5.4.3 The scenario

The last part in the protocol specification consists in describing the execution environments that are considered.

This includes in particular the following important features:

- The attacker's initial knowledge: a set of messages.

- How roles (and more generally composed processes) are composed.

  In typical examples, the roles can run concurrently. Hence a parallel composition operator is used. But other situations might be relevant: there can be some phases/modes as in E-voting protocols, or contract signing, in which case sequential compositions or even conditional composition might be required.

- How roles (and more generally composed processes) can be replicated or dynamically created.

  By "replication of P" we mean that an unbounded number of copies of P may run concurrently. This may, or not, be allowed in the considered scenario and has to be precised. Similar constraints on the number of distinct identities must be specified.

- The hiding or sharing of names.

  This part specifies where the data are generated and how they are inherited. For instance, assume that a role P depend on a parameter k (a key). There is a big difference between `generate` k (`replicate` $P(k)$) and `replicate` (`generate` k $P(k)$). In the former case, each replicated copy of P holds the same key k. In the latter case, each copy of P generates its own key k. The name generation construction not only provides with a name scoping, but also acts as a binder: within that scope, the name can be substituted with a new one. This is necessary in distinguishing instances that generate locally the names.

Later, for the specification of security properties, we will need to distinguish between honest agents, dishonest agents, agents that can be corrupted and so on. The corruption ability may also be specified in the scenario, but also right from the beginning in the categories of names and the different role executions, depending on whether they are played by a corrupted agent or not.

5.5 The specification of security properties

5.5.1 General

In view of the huge variety of security properties (at all levels of abstractions), it seems not possible at the time being to give a general way of describing security properties. Furthermore, there is currently no way to specify (even simple) security properties, independently of the protocol specification language.

Two classes of properties seem mature enough for a formal specification:

- **Trace properties:** they basically express that, in any possible execution, nothing bad can happen. This is the classical case of model checking linear time properties.

- **Equivalence properties:** they state that the attacker cannot get any relevant information on a given data. Such properties are formalised using two experiments, each of which corresponds to the same roles, but to a different scenario. The attacker should be unable to distinguish between the two experiments. Such an indistinguishability property corresponds to the classical notion of observational equivalence in concurrency theory.

We investigate briefly for trace properties some of the main features and relevant information for the specification.

5.5.2 Trace properties

At any time, the global state of the network can be described using the collection of configurations of each role instance, the current scenario, and the sequence S of all messages that have been sent so far on channels that can be eavesdropped.

An *instantaneous property* is a predicate on such global states. For instance, $P(s)$: "a specific message is computable from S by the attacker". Such properties may be referred to as *events*.

A *temporal property* is a combination of instantaneous properties, using temporal modalities. For instance $\phi(s)$: "$P(s)$ never happens", or "Each time $P(s)$ happens, then there was before an event $Q(t)$".

A *trace property* is defined by a quantified temporal property: "for any names $n,\ldots,$ $\phi(s)$". More complex quantifications might be necessary.

Specifying trace properties require then to

1. Specify one or several (parameterized) events;
2. Specify a set of schedules of such events;
3. Specify for which values of the parameters the temporal property must hold.

EXAMPLE Let us specify first informally and then more formally an agreement property. We wish to say that in any instance of the responder role in the Needham-Shroeder protocol, if the variable y of that instance is set to m' at some point, and if both identities that are parameters of that role are honest, then, before that event, a must have generated m' for b.

Formally, we are using two events: $P(b,\ a,\ m)$ and $Q(a,\ b,\ m)$. The first event holds when there is a role instance whose name is B and parameters include the two identities b, a and such that the local variable y is assigned m. The second event hold when there is a role instance whose name is A and parameters include the two identities a, b and such that the local variable x is assigned m.

The temporal property then states that $\neg P(b,\ a,\ m)\ U\ Q(a,\ b,\ m)$: there is no event $P(b,\ a,\ m)$ until an event $Q(a,\ b,\ m)$ occurs. Finally, the authentication (agreement) property states that this holds for any message m and for any two honest identities a, b. Honest (resp. dishonest) identities are here assumed to be distinct name categories.

More complex agreement properties might require the reference to session numbers and are not reported here.

6 Cryptographic protocol assurance levels

6.1 General

The purpose of this International Standards to evaluate the security of cryptographic protocols at a specification-level. This leads to prepare the environment where cryptographic protocols can be used as a part of a whole system, while the part can be regarded as a security-certified black-box.

In ISO/IEC 15408 [13,14,15], when evaluating generic IT products, it is not generally required to prove the security of a standard cryptographic protocol used in the products, while a proprietary cryptographic protocol used in the products is required to show in the framework of ISO/IEC 15408 that they satisfy their security properties described in the Security Target. While the framework of ISO/IEC 15408 does not only require for the designer to show the security feature of the specification, but also requires the correctness of the implementation, it often takes a lot of task. Therefore, this International Standard provides the common basis for verification of security feature of specification. It gives high-level assurance on the specification of a proprietary protocol based on rigorous verification methods so that it enables a proprietary cryptographic protocol to be used in a system as a security-certified black-box as confident as commonly-used standard cryptographic protocols.

NOTE Each of four artifacts in this International Standard corresponds to ISO/IEC 15408 and ISO/IEC 18045 as follows: (1) Specification of the cryptographic protocol can be recognized as a part of TOE Security Functionality (TSF). (2) Adversarial model can be recognized as a part of Operating Environment in which the protocol is executed possibly interacting with an attacker whose ability is specified in the model. (3) Security property can be recognized as a part of formal or informal Security Policy Model (SPM) based on Security Functional Requirement (SFR) which the protocol should satisfy, such as secrecy of exchanged key and authenticity of communicating party, etc. (4) Self-assessment evidence can be recognized as a part of Evidence, that is, what the protocol designer described in the protocol specification as a part of TSF satisfies the security property specified as a part of SPM in the adversarial model specified as a part of Operating Environment.

Followings are three levels of assurance requirements on the design artifacts, which provide increasingly strong guarantees about the security of cryptographic protocols. These levels have associated requirements.

Table 1 — Cryptographic protocol assurance levels

Protocol Assurance Level	**PAL1**	**PAL2**	**PAL3**	**PAL4**
Protocol Specification	**PPS_SEMIFORMAL** Semiformal description of protocol specification.	**PPS_FORMAL** Formal description of protocol specification.	**PPS_ MECHANIZED** Formal description of protocol specification in a tool-specific specification language, whose semantics is mathematically defined.	
Adversarial Model	**PAM_INFORMAL** Informal description of adversarial model.	**PAM_ FORMAL** Formal description of adversarial model.	**PAM_ MECHANIZED** Formal description of adversarial model in a tool-specific specification language, whose semantics is mathematically defined.	
Security Property	**PSP_INFORMAL** Informal description of security property	**PSP_ FORMAL** Formal description of security property.	**PSP_ MECHANIZED** Formal description of security property in a tool-specific specification language, whose semantics is mathematically defined.	
Self-assessment Evidence	**PEV_ARGUMENT** Informal argument that the specification of the cryptographic protocol in its adversarial model achieves and satisfies its objectives and properties.	**PEV_HANDPROVEN** Mathematically formal paper-and-pencil proof verified by human that the specification of the cryptographic protocol in its adversarial model achieves and satisfies its objectives and properties.	**PEV_BOUNDED** Tool-aided bounded verification that the specification of the cryptographic protocol in its adversarial model achieves and satisfies its objectives and properties.	**PEV_UNBOUNDED** Tool-aided unbounded verification that the specification of the cryptographic protocol in its adversarial model achieves and satisfies its objectives and properties.

6.2 Protocol Assurance Level 1

a) Security relevant parts of the protocol shall be specified in a semiformal language, e.g., as a sequence of message exchanges.

b) Informal specification of the adversarial model.

c) Informal specification of the security properties.

d) Informal argument of why the protocol has the specified properties.

6.3 Protocol Assurance Level 2

a) Security relevant parts of the protocol shall be specified in a formal language.

b) Adversarial model specified in a formal language.

c) Properties specified in a formal language.

d) Mathematically formal paper-and-pencil proof of why the protocol has the specified properties.

6.4 Protocol Assurance Level 3

a) Security relevant parts of the protocol shall be specified in a formal tool-specific specification language. The protocol model may bound the number of role instances to some specified value.

b) Adversarial model specified in a formal language.

c) Properties specified in a formal language.

d) Self-assessment evidence for verification (solving the model-checking problem) by a model checking or theorem proving tool.

6.5 Protocol Assurance Level 4

a) Security relevant parts of the protocol shall be specified in a formal tool-specific specification language. The protocol model shall allow for unboundedly many number of role instances

b) Adversarial model specified in a formal language.

c) Properties specified in a formal language.

d) Self-assessment evidence for verification either by a model checking tool or by a theorem proving tool. For model checking, procedures should be employed that can handle the infinite state space that arises in unbounded verification.

6.6 Difference among Protocol Assurance Levels

The difference between PAL1 and PAL2 is whether all aspects of the protocol description, such as the specification, security property, and adversarial model, are formally described or not. If these are not sufficiently formal, a rigorous analysis is not possible and the designer cannot search for attacks or construct correctness proofs. At best, the designer can search for typical weakness and evaluate the protocol with respect to those attacks that she has thought of. Hence, PAL1 gives only minimal guarantees about the protocols security. However, PAL1 might be sufficient for some closed network environment, such as a company intranet, lacking committed adversaries.

In contrast, in PAL2 and higher levels, the protocol designer provides a formal specification. This makes it possible to search for attacks rigorously or construct correctness proofs.

The difference between PAL2 and PAL3 is whether the security of the protocol is verified by hand or by mechanized tools. At least in theory, protocol verification can be carried out by hand proofs, using paper and pencil. However, given the substantial amount of detail typically involved in cryptographic protocol verification, hand proofs could be error prone and confidence in the results is substantially increased by using mechanized tools such as model checkers and theorem provers. Thus, proofs only with paper-and-pencil are treated as a lower assurance level (i.e. PAL2) than mechanized proof in this International Standard. PAL2 is generally effective for only simple protocols in open network environment such as the Internet.

In PAL3, the protocol designer provides a protocol specification in a formal tool-specific specification language. Thus she can capture all traces consistent with the specification within some bound specified for the verification. Designers are typically poor at anticipating all possible (interleaved) traces and hence these traces will usually include complex ones, not considered in advance by the protocol designer. PAL3 generally gives reasonable guarantees that there does not exist any other successful adversary at all within some bound on the number of protocol sessions. PAL3 can be recommended for open network environment.

The difference between PAL3 and PAL4 is whether or not the analysis (and hence the evidence presented) is for unbound verification or not. Verification in PAL3, is bounded and thus the designer cannot prove security of her protocol for complex attacks which lie outside the bound. In contrast, PAL3 gives strong guarantees on that no successful (symbolic, Dolev-Yao) adversary exists, even allowing for unbounded numbers of sessions. With unbounded verification, a protocol designer can prove security of her protocol against all adversaries, even those willing to carry out complex and expensive attacks. PAL4 is effective for critical information systems, such as those providing social infrastructures or financial systems.

6.7 Corresponding assurance levels in ISO/IEC 15408

The purpose of this International Standard is to specify the method to evaluate the security of cryptographic protocols at a specification-level. First, it should be emphasized on that this International Standard is useful even in evaluating specification-only cryptographic protocols independently from the whole system where the protocols are implemented. Second, this International Standard is designed to be useful in evaluating cryptgraphic protocols in critical IT systems which is often the target of ISO/IEC 15408 cerificaton.

NOTE In a case of using this International Standard jointly with ISO/IEC 15408 to evaluate total security of IT systems including cryptographic protocols, Table 2 gives the envisioned corresondence between Evaluation Assurance Levels (EAL) in ISO/IEC 15408 of the IT system and Protocol Assurance Levels (PAL) in this International Standard. In this case, the developper should identify a subset of cryptographic protocols in TOE as a set of target protocols to be evaluated in this International Standard. This set of target protocols should cover all critical cryptographic protocols in TOE. Cryptographic protocols which exchange important information over an open network or are used as a critical social infrastructure are examples of the prospective target protocols.

Table 2 (Informative) — Envisioned correspondence between EAL and PAL

ISO/IEC 15408 EAL	ISO/IEC 29128 PAL	
	Target Protocols in TOE	Other Protocols in TOE
EAL 7	PAL 4	
EAL 6	PAL 3	
EAL 5		
EAL 4	PAL 2	
EAL 3		
EAL 2	PAL 1	
EAL 1		

7 Security Assessment and Verification

7.1 Protocol specification

7.1.1 PPS_SEMIFORMAL

7.1.1.1 Objectives

Protocol specification in PPS_SEMIFORMAL shall specify the security relevant parts of the protocol in a semiformal language, usually as a sequence of message exchanges. Annex 1, “Guidelines for Cryptographic Protocol Design”, describes relevant issues when a protocol designer designs or evaluates a protocol for PPS_SEMIFORMAL. A protocol designer shall take these issues into account in designing the secure protocol specification.

7.1.1.2 Developer action elements

The developer shall provide specification of target cryptographic protocol written in semi-formal language.

7.1.1.3 Content and presentation elements

The protocol specification shall describe sequences of the protocol, which include cryptographic operations executed by all protocol participants, required information of these cryptographic operations and protocol message.

7.1.1.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

7.1.2 PPS_FORMAL

7.1.2.1 Objectives

Protocol specification in PPS_FORMAL shall specify the security relevant parts of the protocol in a formal language. The semantics of the language shall be explicitly or implicitly defined.

7.1.2.2 Developer action elements

The developer shall provide PPS_SEMIFORMAL.

The developer shall provide specification of target cryptographic protocol written in a formal language.

7.1.2.3 Content and presentation elements

The formal protocol specification shall describe all necessary items in Clause5.3 such as messages, observing messages, algebraic properties, and protocol roles written in a formal language. This formal protocol specification shall conform to PPS_SEMIFORMAL.

7.1.2.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

7.1.3 PPS_MECHANIZED

7.1.3.1 Objectives

Protocol specification in PPS_MECHANIZED shall specify the security relevant parts of the protocol in a formal tool-specific specification language.

NOTE 1 Protocols can be specified in a tool-specific specification language as a protocol automaton representation (such as Mealy Machines) for the different principals, or specified as a strand-style notation [4], where one lists, for each role, the messages sent and received.

NOTE 2 Protocols can be specified in general predicate logic language as an inductively defined set where the set formalizes the possible actions (e.g., communication events) arising from principals following their protocol roles as well as an intruder. Protocols can also be specified with notation based on processes, transition systems, or strands, where the formalization can be in terms of the local actions of the different principals (e.g., their individual processes or local transition systems) or the global transition system.

7.1.3.2 Developer action elements

The developer shall provide PPS_SEMIFORMAL.

The developer shall provide a specification of the target cryptographic protocol written in a formal tool-specific specification language.

The developer shall provide information about the language which is used for describe specification of the target protocol.

7.1.3.3 Content and presentation elements

The formal protocol specification shall describe all necessary items in Clause 5.3 such as messages, observing messages, algebraic properties, and protocol roles written in a formal tool-specific specification language. This formal protocol specification shall conform to PPS_SEMIFORMAL. Information about the tool-specific specification language shall include materials such as reference manuals and papers.

7.1.3.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

7.2 Adversarial model

7.2.1 PAM INFORMAL

7.2.1.1 Objectives

Adversarial model for PAM_INFORMAL shall specify the intruder model, properties of communication channel and scenario typically in an informal language.

7.2.1.2 Developer action elements

The developer shall provide a statement of adversarial model.

7.2.1.3 Content and presentation elements

The statement of adversarial model shall describe all necessary items in Clause 5.4 such as network specification, ability of attacker and scenario.

7.2.1.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

7.2.2 PAM_FORMAL

7.2.2.1 Objectives

Adversarial model for PAM_FORMAL shall specify the intruder model and properties of communication channel in a formal language. Semantics of the language shall be explicitly or implicitly defined. This may be the Dolev-Yao[3] intruder or some variant thereof. The Dolev-Yao intruder model is a typical operating environment in which the network is under full control of the intruder, that is, any message over the network can be learnt, falsified, and created by the intruder. Variants may include weaker intruders such as a passive intruder who can only eavesdrop on the network, or a wireless intruder who can both eavesdrop and create any message but cannot deflect a message with learning the message. Variants may also arise when additional properties are modeled of cryptographic operators (e.g., algebraic properties [1, 2]).

7.2.2.2 Developer action elements

The developer shall provide PAM_INFORMAL.

The developer shall provide a statement of adversarial model in a formal language.

7.2.2.3 Content and presentation elements

The statement of adversarial model shall describe all necessary items in Clause 5.4 such as network specification, ability of attacker and scenario written in a formal language. This formal adversarial model shall conform to PAM_INFORMAL.

7.2.2.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

7.2.3 PAM_MECHANIZED

7.2.3.1 Objectives

Adversarial model for PAM_MECHANIZED shall specify the intruder model and properties of communication channel in a tool-specific specification formal language. This may be the Dolev-Yao[3] intruder or some variant thereof. The Dolev-Yao intruder model is a typical operating environment in which the network is under full control of the intruder, that is, any message over the network can be learnt, falsified, and created by the intruder. Variants may include weaker intruders such as a passive intruder who can only eavesdrop on the network, or a wireless intruder who can both eavesdrop and create any message but cannot deflect a message with learning the message. Variants may also arise when additional properties are modeled of cryptographic operators (e.g., algebraic properties [1, 2]).

7.2.3.2 Developer action elements

The developer shall provide PAM_INFORMAL.

The developer shall provide a statement of adversarial model in a tool-specific specification language.

The developer shall provide information about language which is used for describe adversarial model of the target protocol.

7.2.3.3 Content and presentation elements

The statement of adversarial model shall describe all necessary items in Clause 5.4 such as network specification, ability of attacker and scenario written in a formal tool-specific specification language. This formal adversarial model shall conform to PAM_INFORMAL.

Information about the tool-specific specification language shall include materials such as reference manuals and papers.

7.2.3.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

NOTE 1 Adversarial model can be specified in tool-specific specification language. For efficiency reasons, most tools “hard-wire” this intruder model into their tool, e.g., via specialized calculi for constraint solving. If one uses a general-purpose model checker, one must explicitly represent the intruder as a process.

NOTE 2 Adversarial model can be specified in general predicate logic language as an inductively defined set as part of the protocol specification. The adversarial model can also be specified as an intruder formalization integrated within verification tools.

7.3 Security properties

7.3.1 General

Properties φ specify requirements on the behavior of the protocol. For the vast majority of Formal Methods, the model M constitutes a transition system, describing the system states and the transitions between states. In this setting, a property φ is typically either a state-property, i.e., some invariant that should hold for all reachable system states or a temporal property describing valid sequences of states (or system events). Standard examples of security properties are secrecy properties or authentication properties. A secrecy (or confidentiality) property is generally a state invariant and says that some set S of data items (e.g., keys and nonces) is never obtained by the attacker in an unencrypted form. Authentication properties include both message-origin authentication (that a message supposedly sent by a party actually was sent by the party) and entity authentication (roughly, that a given principal is currently participating in the protocol in some stated role).

7.3.2 PSP_INFORMAL

7.3.2.1 Objectives

Security properties for PSP_INFORMAL shall specify informal specification of the security properties in an informal language.

7.3.2.2 Developer action elements

The developer shall provide statement about the security property which the protocol should achieve.

7.3.2.3 Content and presentation elements

The statement about the security property shall cover all properties which the protocol should achieve.

7.3.2.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

7.3.3 PSP_FORMAL

7.3.3.1 Objectives

Security properties for PSP_FORMAL shall specify requirements on the behavior of the protocol in a formal language. Semantics of the language shall be explicitly or implicitly defined.

7.3.3.2 Developer action elements

The developer shall provide PSP_INFORMAL.

The developer shall provide statement about the security property which the protocol should achieve. This shall be written in formal language.

7.3.3.3 Content and presentation elements

The statement about security property shall cover all properties which the protocol should achieve written in a formal language in a way described in Clause 5.5. This formal security property shall conform to PSP_INFORMAL.

7.3.3.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

7.3.4 PSP_MECHANIZED

7.3.4.1 Objectives

Security properties for PSP_MECHANIZED shall specify requirements on the behavior of the protocol in a formal tool-specific specification language.

7.3.4.2 Developer action elements

The developer shall provide PSP_INFORMAL.

The developer shall provide statement about security property which the protocol should achieve. This shall be written in formal tool-specific specification language.

The developer shall provide information about language which is used for describe security properties of the target protocol.

7.3.4.3 Content and presentation elements

The statement about security property shall cover all properties which the protocol should achieve written in a formal tool-specific specification language in a way described in Clause 5.5. This formal security property shall conform to PSP_INFORMAL.

Information about tool-specific specification language shall include materials such as reference manual and papers.

7.3.4.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

NOTE Verification tools such as state-of-the-art model-checkers provide a collection of predefined predicates to formalise typical security properties such as secrecy and various notions of authentication. Hence, with such verification tools, a security property specification consists of selecting and instantiating appropriate security predicates, for example, indicating which terms should remain secret, or which values should be agreed upon by which principals.

7.4 Self-assessment evidence for verification

7.4.1 General

Self-assessment evidence of verification is an evidence that the protocol designer solves the model-checking problem of demonstrating that the model M has the property φ, i.e., M satisfies φ. There are a variety of different Formal Methods capable of establishing or refuting that M satisfies φ.

7.4.2 PEV_ARGUMENT

7.4.2.1 Objectives

The evidence for verification for PEV_ARGUMENT consists of informal argument of why the protocol has the specified properties.

7.4.2.2 Developer action elements

The developer shall provide information how the protocol specification fulfills the security property in the adversarial model.

7.4.2.3 Content and presentation elements

The information shall describe the reason in a verifiable way why the protocol specification assures the security property in the adversarial model.

7.4.2.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

7.4.3 PEV_HANDPROVEN

7.4.3.1 Objectives

The evidence for verification for PEV_HANDPROVEN consists of mathematically formal paper-and-pencil proof of why the protocol has the specified properties.

7.4.3.2 Developer action elements

The developer shall provide information how the protocol specification fulfills the security property in the adversarial model.

7.4.3.3 Content and presentation elements

The information shall describe the reason in a verifiable way why the protocol specification assures the security property in the adversarial model.

7.4.3.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

7.4.4 PEV_BOUNDED

7.4.4.1 Objectives

The evidence for verification for PEV_BOUNDED consists of the protocol specification and security properties and (if required) the adversarial model used by a verification tool such as model checkers along with additional parameters. The most important parameter is to provide a bound on the number of role instances. Different model checkers provide different ways of doing this. The evidence is input to the tool. If a protocol satisfies declared properties, the tools simply report this. If the protocol fails to satisfy the properties, the tools should provide informative output such as a trace or message sequence chart.

7.4.4.2 Developer action elements

The developer shall provide a proof of how the protocol specification of PPS_MECHANIZED fulfills the security property of PSP_MECHANIZED on the adversarial model of PAM_MECHANIZED for a bounded number of role instances.

The developer shall provide a statement that the verification tool used is sound.

7.4.4.3 Content and presentation elements

The proof shall describe results of verification tool in such a way that they are repeatable. The proof shall clearly specify that number of role instances with which the security of the protocol is assured.

The statement shall describe the correctness of the verification tool used and the soundness of the proof.

7.4.4.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

7.4.5 PEV_UNBOUNDED

7.4.5.1 Objectives

The evidence for verification for PEV_UNBOUNDED is an evidence that the protocol designer solves the model-checking problem of demonstrating that the model M has the property φ for an unbounded number of inter-leaving role instances.

This evidence shall be constructed with the aid of a verification tool. The evidence required for verification depends on the verification tool. Protocol designers may choose state-of-the-art verification tools such as theorem provers and model-checkers. Theorem provers for first or higher-order logic usually take as input a script of commands. These commands include declarations of definitions and lemmas as well as commands for constructing proofs themselves. The proofs are built from inference rules or tactics, which are programs that construct proofs. In the verification process, the script is input to the theorem prover. If any of the commands fail, then the verification fails. If the proof script runs successfully, then all theorems stated in the script, and therefore proven about the protocol, are valid.

The validity of the final results depends on the soundness of the logic used, its correct implementation in the theorem prover, and that all extensions done to the logic (by adding definitions, rules, etc.) were done in a way that preserves the logic's consistency. Protocol designers shall provide information regarding the validity of the results.

7.4.5.2 Developer action elements

The developer shall provide a proof how the protocol specification of PPS_MECHANIZED fulfills the security property of PSP_MECHANIZED on the adversarial model of PAM_MECHANIZED for unbounded number of role instances.

The developer shall provide a statement that the verification tool used is sound.

7.4.5.3 Content and presentation elements

The proof shall describe results of verification tool in such a way that they are repeatable.

The proof shall indicate that the security of the protocol is assured with unbounded number of role instances.

The statement shall describe the correctness of the verification tool used and the soundness of the proof.

7.4.5.4 Evaluator action elements

The evaluator shall confirm that the information provided meets all requirements for content and presentation.

8 Common Methodology for Cryptographic Protocols Security Evaluation

8.1 Introduction

This clause shows common methods for cryptographic protocol evaluation according to this International Standard.

8.2 Protocol specification evaluation

8.2.1 Evaluation of sub-activity (PPS_SEMIFORMAL)

8.2.1.1 Input

The evaluation evidence for this sub-activity is:

- The specification of target cryptographic protocol written in semi-formal language.

8.2.1.2 Action

The evaluator shall confirm that the protocol specification describe sequences of the protocol, which include cryptographic operations executed by all protocol participants, required information of these cryptographic operations and protocol message.

8.2.2 Evaluation of sub-activity (PPS_FORMAL)

8.2.2.1 Input

The evaluation evidence for this sub-activity is:

- The PPS_SEMIFORMAL.
- The specification of target cryptographic protocol written in a formal language.

8.2.2.2 Action

The evaluator shall confirm that the formal protocol specification describes all necessary items in Clause 5.3 such as messages, observing messages, algebraic properties, and protocol roles written in a formal language.

The evaluator shall confirm that this formal protocol specification conforms to PPS_SEMIFORMAL.

8.2.3 Evaluation of sub-activity (PPS_MECHANIZED)

8.2.3.1 Input

The evaluation evidence for this sub-activity is:

- The PPS_SEMIFORMAL.
- The specification of target cryptographic protocol written in a formal tool-specific specification language.
- The information about language which is used for describe specification of the target protocol.

8.2.3.2 Action

The evaluator shall confirm that the formal protocol specification describes all necessary items in Clause 5.3 such as messages, observing messages, algebraic properties, and protocol roles written in a formal tool-specific specification language.

The evaluator shall confirm that this formal protocol specification conforms to PPS_SEMIFORMAL.

The evaluator shall confirm that Information about tool-specific specification language includes materials such as reference manual and papers

8.3 Adversarial model evaluation

8.3.1 Evaluation of sub-activity (PAM INFORMAL)

8.3.1.1 Input

The evaluation evidence for this sub-activity is:

- The statement of adversarial model.

8.3.1.2 Action

The evaluator shall confirm that the statement of adversarial model describes all necessary items in Clause 5.4 such as network specification, ability of attacker and scenario.

8.3.2 Evaluation of sub-activity (PAM_FORMAL)

8.3.2.1 Input

The evaluation evidence for this sub-activity is:

- The PAM_INFORMAL.
- The statement of adversarial model in a formal language.

8.3.2.2 Action

The evaluator shall confirm that the statement of adversarial model describes all necessary items in Clause 5.4 such as network specification, ability of attacker and scenario written in a formal language.

The evaluator shall confirm that this formal adversarial model conforms to PAM_INFORMAL.

8.3.3 Evaluation of sub-activity (PAM_MECHANIZED)

8.3.3.1 Input

The evaluation evidence for this sub-activity is:

- The PAM_INFORMAL.
- The statement of adversarial model in a formal tool-specific specification language.
- The information about language which is used for describe adversarial model of the target protocol.

8.3.3.2 Action

The evaluator shall confirm that the statement of adversarial model describes all necessary items in Clause 5.4 such as network specification, ability of attacker and scenario written in a formal tool-specific specification language.

The evaluator shall confirm that this formal adversarial model conforms to PAM_INFORMAL.

The evaluator shall confirm that information about tool-specific specification language includes materials such as reference manual and papers.

8.4 Security properties evaluation

8.4.1 Evaluation of sub-activity (PSP_INFORMAL)

8.4.1.1 Input

The evaluation evidence for this sub-activity is:

- The statement about security property which the protocol should achieve.

8.4.1.2 Action

The evaluator shall confirm that the statement about security property covers all properties which the protocol should achieve.

8.4.2 Evaluation of sub-activity (PSP_FORMAL)

8.4.2.1 Input

The evaluation evidence for this sub-activity is:

- The PSP_INFORMAL.
- The statement about security property which the protocol should achieve. This shall be written in a formal language.

8.4.2.2 Action

The evaluator shall confirm that the statement about security property covers all properties which the protocol should achieve written in a formal language in a way described in Clause 5.5.

The evaluator shall confirm that this formal security property conforms to PSP_INFORMAL.

8.4.3 Evaluation of sub-activity (PSP_MECHANIZED)

8.4.3.1 Input

The evaluation evidence for this sub-activity is:

- The PSP_INFORMAL.
- The statement about security property which the protocol should achieve. This shall be written in a formal tool-specific specification language.
- The information about language which is used for describe security properties of the target protocol.

8.4.3.2 Action

The evaluator shall confirm that the statement about security property covers all properties which the protocol should achieve written in a formal tool-specific specification language in a way described in Clause 5.5.

The evaluator shall confirm that this formal security property conforms to PSP_INFORMAL.

The evaluator shall confirm that the information about tool-specific specification language includes materials such as reference manual and papers.

8.5 Self-assessment evidence evaluation

8.5.1 Evaluation of sub-activity (PEV_ARGUMENT)

8.5.1.1 Input

The evaluation evidence for this sub-activity is:

- The information how the protocol specification fulfills the security property in the adversarial model.

8.5.1.2 Action

The evaluator shall confirm that the information describes the reason in a verifiable way why the protocol specification assures the security property in the adversarial model.

8.5.2 Evaluation of sub-activity (PEV_HANDPROVEN)

8.5.2.1 Input

The evaluation evidence for this sub-activity is:

- The mathematically formal paper-and-pencil proof how the protocol specification fulfills the security property in the adversarial model.

8.5.2.2 Action

The evaluator shall confirm that the proof describes the reason in a verifiable way why the protocol specification assures the security property in the adversarial model.

8.5.3 Evaluation of sub-activity (PEV_BOUNDED)

8.5.3.1 Input

The evaluation evidence for this sub-activity is:

- The proof how the protocol specification of PPS_MECHANIZED fulfills the security property of PSP_MECHANIZED on the adversarial model of PAM_MECHANIZED for bounded number of role instances.

- The statement that the verification tool used is sound.

8.5.3.2 Action

The evaluator shall confirm that the proof describes results of verification tool in such a way that they are repeatable.

The evaluator shall confirm that the proof clearly specifies that number of role instances with which the security of the protocol is assured.

The evaluator shall confirm that the statement describes the correctness of the verification tool used and the soundness of the proof.

8.5.4 Evaluation of sub-activity (PEV_UNBOUNDED)

8.5.4.1 Input

The evaluation evidence for this sub-activity is:

- The proof how the protocol specification of PPS_MECHANIZED fulfills the security property of PSP_MECHANIZED on the adversarial model of PAM_MECHANIZED for unbounded number of role instances.

- The statement that the verification tool used is sound.

8.5.4.2 Action

The evaluator shall confirm that the proof describes results of verification tool in such a way that they are repeatable.

The evaluator shall confirm that the proof indicates that the security of the protocol is assured with unbounded number of role instances.

The evaluator shall confirm that the statement describes the correctness of the verification tool used and the soundness of the proof.

Annex A
(informative)

Guidelines for Cryptographic Protocol Design

This informative annex gives known guidelines for the design of cryptographic protocols. These guidelines will help for a cryptographic protocol designer to avoid typical errors. Note that a protocol designed along with the principles is not necessarily compliant with this International Standard. The designed protocol should be verified separately to be compliant.

This clause gives the design guidelines in [5]. Similar guidelines are also given in [6].

1 Every message should say what it means the interpretation of the message should depend only on its content. It should be possible to write down a straightforward English sentence describing the content - though if there is a suitable formalism available that is good too.

2 The conditions for a message to be acted upon should be clearly set out so that someone reviewing a design can see whether they are acceptable or not.

3 If the identity of a principal is essential to the meaning of a message it is prudent to mention the principal's name explicitly in the message.

4 Be clear about why encryption is being done. Encryption is not wholly cheap and not asking precisely why it is being done can lead to redundancy. Encryption is not synonymous with security and its improper use can lead to errors.

5 When a principal signs material that has already been encrypted it should not be inferred that the principal knows the content of the message. On the other hand it is proper to infer that the principal that signs a message and then encrypts it for privacy knows the content of the message.

6 Be clear what properties you are assuming about nonces. What might do for ensuring temporal succession might not do for ensuring association and perhaps association is best established by other means.

7 The use of a predictable quantity such as the value of a counter can serve in guaranteeing newness through a challenge-response exchange. But if a predictable quantity is to be effective it should be protected so that an intruder cannot simulate a challenge and later replay a response.

8 If timestamps are used as freshness guarantees by reference to absolute time then the difference between local clocks at various machines must be much less than the allowable age of a message

deemed to be valid. Furthermore the time maintenance mechanism everywhere becomes part of the trusted computing base.

9 A key can have been used recently for example to encrypt a nonce yet be quite old and possibly compromised. Recent use does not make the key look any better than it would otherwise.

10 If an encoding is used to present the meaning of a message then it should be possible to tell which encoding is being used. In the common case where the encoding is protocol dependent, it should be possible to deduce that the message belongs to this protocol, and in fact to a particular run of the protocol, and to know its number in the protocol.

11 The protocol designer should know which trust relations his protocol depends on and why the dependence is necessary. The reasons for particular trust relations being acceptable should be explicit though they will be founded on judgment and policy rather than on logic.

Annex B
(informative)

Example of formal specification

B.1 Symbolic specification of security protocols

Here we describe a way of formal specification of security protocols along the guidelines given in Clause 5. Although it still remains abstract to a certain degree, the description below demonstrates a way of using formal methods in the verification of protocols. Furthermore, we explain each step with a concrete example using the simplified Needham-Schroeder public-key protocol.

Our description style follows the syntax of Cas Cremer's Scyther [9] because of its simple and intuitive use for the specification of security protocols. However, the description below corresponds also to the basic concepts of Bruno Blanchet's ProVerif [10], one of the leading tools for the automatic symbolic verification of security protocols.

B.1.1 Abstraction level

The protocols are specified at the most abstract level, i.e., messages are terms constructed from symbols and the attacker is modeled as a formal process.

B.1.1.1 Languages

A formal language $\mathcal{L} = (\mathcal{V}, \mathcal{C}, \mathcal{R}, \mathcal{F})$ for the Needham-Schroeder public-key protocol consists of a set $\mathcal{V}$ of variables to store received messages, a set $\mathcal{C}$ of local constant symbols for nonces, a set $\mathcal{R}$ of role name symbols, a set $\mathcal{F}$ of function symbols for two key-generating functions pk and sk for public or private key generation, respectively, a binary pairing function $(\cdot,\cdot)$, two unary functions π_1, π_2 for decompositions of pairs, two binary functions `enc` and `dec` for encryption and decryption, respectively.

It is assumed that the cryptographic primitives such as `enc` and `dec` are perfect. Only their relevant properties are assumed to be known. We impose no restriction on the construction of a term, i.e., all terms that are built with the function symbols following the arity restrictions are valid messages: A term is a variable $v \in \mathcal{V}$, a nonce $N \in \mathcal{C}$, a role name $r \in \mathcal{R}$, or a compound term using a function symbol.

We could have used some restrictions that some arguments are restricted to some classes. Instead we assume that terms are distinguishable according to their types: role names, nonces, keys, encryption, tuples, etc. This means, e.g., that only keys can be used to encrypt a message, otherwise the terms will be ignored or discarded directly.

B.1.1.2 Algebraic properties

Besides the syntactic equality of terms, the symbolic-level specification of message equality includes the following basic equations:

$$\begin{aligned} \mathbf{dec}(\mathrm{sk}(r),\mathbf{enc}(\mathrm{pk}(r),m)) &= m \\ \pi_i(m_1,m_2) &= m_i \; i \in \{1,2\} \end{aligned}$$

We remark that, in general, adding algebraic properties of ⊕, *exclusive or*, with a homomorphic hash function to the Dolev-Yao attacker model

$$\begin{aligned} (x \oplus y) \oplus z &= x \oplus (y \oplus z) \\ x \oplus y &= y \oplus x \\ 0 \oplus x &= x \\ x \oplus 0 &= x \\ x \oplus x &= 0 \end{aligned}$$

$$\begin{aligned} h(x \oplus y) &= h(x) \oplus h(y) \\ \mathbf{enc}(k, x \oplus y) &= \mathbf{enc}(k,x) \oplus \mathbf{enc}(k,y) \end{aligned}$$

weakens the perfect cryptographic assumption [11].

We also remark that the current state of Scyther or ProVerif cannot deal with the exclusive or. On the other hand, there are some researches on how to deal with the exclusive or for automatic verification tools [12].

B.1.2 Protocol specifications

Given a language $\mathcal{L} = (\mathcal{V}, \mathcal{C}, \mathcal{R}, \mathcal{F})$, a protocol P describes the behaviour of each of the roles such as initiator, responder, key server, etc. In the specification, the behaviour of each role is formalized as a transition system describing how to create messages, how to react to the received messages, and how to manipulate them. We assume that all the communications are made through a single public channel c.

B.1.2.1 Sending and receiving events

We use two ternary predicate symbols exist: **`Send`** for sending messages and **`Receive`** for receiving messages:

$$\mathtt{Send}(i,r,m) \text{ for } i \text{ sends the message } m \text{ to } r.$$
$$\mathtt{Receive}(i,r,m) \text{ for } r \text{ receives the message } m \text{ from } i.$$

B.1.2.2 Role specifications

A protocol consists of role specifications for each role. A role specification describes the behaviour of a role, which is represented as a finite sequence of sending and receiving events. An initial knowledge also belongs to the role specification. The initial knowledge of a role contains the names of other roles, public keys of all roles, his own private key, etc.

The Needham-Schroeder public-key protocol contains two specifications, one for the initiator, the other for the responder.

- For the initiator: The agent taking the initiator role I knows the identity of the agent taking the responder role R and its public key $\mathrm{pk}(R)$ besides his own private key $\mathrm{sk}(I)$ and can create a nonce N_i.

$$\mathtt{Send}(I,R,\{N_i,I\}_{\mathrm{pk}(R)});$$
$$\mathtt{Receive}(R,I,\{N_i,x\}_{\mathrm{pk}(I)});$$
$$\mathtt{Send}(I,R,\{x\}_{\mathrm{pk}(R)})$$

- For the responder: The agent taking the responder role R knows the identity of the agent taking the initiator role I and its public key $\mathrm{pk}(I)$ besides his own private key $\mathrm{sk}(R)$ and can create a nonce N_r.

$$\mathtt{Receive}(I,R,\{y,I\}_{\mathrm{pk}(R)});$$
$$\mathtt{Send}(R,I,\{y,N_r\}_{\mathrm{pk}(I)});$$
$$\mathtt{Receive}(I,R,\{N_r\}_{\mathrm{pk}(R)})$$

B.1.2.3 Computation

Before any event is executed, each agent performs some computations, while the state changes after each event execution. What exactly the computation does, depends on the protocol model. Below are two standard examples of computation.

- Pattern matching: It can be assumed that any agent in a role r could see the pattern of any message: nonces, agent names, session keys, pairs, encrypted messages, etc. Messages which do not follow the pattern matching will be ignored, forwarded, or discarded directly.

- Knowledge inference: Based on the pattern-matching, any agent including the attacker can infer new knowledge from his initial knowledge together with the previously received messages. The knowledge of an agent is obtained from his initial knowledges or received messages using his capability to manipulate them like composition, discomposition, encryption, decryption, etc.

B.2 State transitions

In this example, the roles can run events concurrently and each event can be repeated unlimitedly many times.

B.2.1 Attacker model

The network can be partially or completely under control of an attacker. Based on his knowledge, he can e.g. catch, eavesdrop, or fake messages. He can also interrupt or disturb the protocol running.

The initial knowledge $\mathcal{K}_A{}^0$ of an agent A in a role consists of e.g. the names and public keys of all agents and his private key of his role. The initial attacker knowledge $\mathcal{K}_I{}^0$ consists of the initial knowledge of all untrusted agents including their private keys. The knowledge of an agent including the attacker will grow during the running of the protocol whenever he receives or catch messages.

B.2.2 Configuration state

The configuration state at some point during running a protocol P is composed of the local attacker knowledge and the local knowledge of every possible agent A_n, where n varies over natural numbers. The list of agents is made infinite such that it reflects the fact that the attacker could initiate new session at any step and perform unlimited sessions. In the initial state, every agent is in his initial state, i.e. his initial knowledge and initial control state. If an agent $A_n$is not active yet, then his initial knowledge is empty.

The state transition rules of a protocol P describe how configuration states changes during the protocol running. Below are three rules which constitute the operational semantics of the protocol P.

— **New instances** If an agent performs a new instance event, then he adds a new role instance to his state. This is based on the assumption that unlimited number of role instantiation is possible. The agent performs this role gets then the initial knowledge assigned to the role. If the agent is compromised, then he shares all the knowledge with the attacker.

— **Sending event** If an agent performs a sending event, the sent message m is added to the attacker knowledge. Then he performs some computations, and depending on the computation result, he moves to some state where he is waiting for another sending or receiving event, or stops.

— **Receiving event** If an agent performs a receiving event, the agent performs some computations. Depending on the computation result, he moves to some state where he is waiting for another sending or receiving event, or stops. The computations include e.g. the readability test. If the received message m passes the readability test, the message will be added to the knowledge of the agent.

B.2.3 Traces

A state transition is the conclusion of finitely many applications of the rules above, starting from the initial state. A trace of a protocol P is the description of any possible state transition starting from the initial state:

$$\left(\mathcal{K}_I^0, \langle\mathcal{K}_{A_n}^0\rangle_n\right) \xrightarrow{m_1} \cdots \xrightarrow{m_\ell} \left(\mathcal{K}_I^\ell, \langle\mathcal{K}_{A_n}^\ell\rangle_n\right) \xrightarrow{m_{\ell+1}} \left(\mathcal{K}_I^{\ell+1}, \langle\mathcal{K}_{A_n}^{\ell+1}\rangle_n\right) \xrightarrow{m_{\ell+2}} \cdots$$

Here ℓ denotes the ℓ-th transition step and m_ℓ is the exchanged message at the ℓ-th transition. $\langle\mathcal{K}_{An}{}^{\ell}\rangle_n$ and $\mathcal{K}_I{}^{\ell}$ stand for the local knowledge of the agent A and of the attacker I, respectively, at the ℓ-th step. $\langle\mathcal{K}_{An}{}^{\ell}\rangle_n$ is an abbreviation for the infinite list of $\mathcal{K}_{An}{}^{\ell}$, where n varies over natural numbers. The relationship between m_ℓ and m_i, $i < m$, is decided by the protocol specification.

B.3 Trace properties

Among many security properties we illustrate here two trace properties: secrecy and authentication.

B.3.1 Secrecy

Secrecy expresses that certain information cannot be revealed to any other agent or the attacker except the honest agents who have run the protocol, even though the protocol is executed in an untrusted network. More formally, a protocol P satisfies secrecy of a message m among some honest agents $A_{n1},\cdots, A_{np}$ if and only if in an arbitrary trace, m cannot be inferred from the knowledge of anybody else, i.e.,

$$\mathcal{K}_I^\ell \nvdash m \quad \text{and} \quad \mathcal{K}_A^\ell \nvdash m$$

for any ℓ, where A is not one of $A_{n1},\cdots, A_{np}$ and $\mathcal{K} \vdash m$ denotes that m can be inferred from the local knowledge $\mathcal{K}$. In case of the Needham-Schroeder public-key protocol, the secrecy question is if $\mathcal{K}_I{}^{\ell} \nvdash N_r$ holds for any moment ℓ in a trace if the roles I and R are performed by two honest agents. The man-in-the-middle-attack, found by G. Lowe, against the Needham-Schroeder public-key protocol demonstrates that the secrecy N_b generated by an honest agent B is not guaranteed.

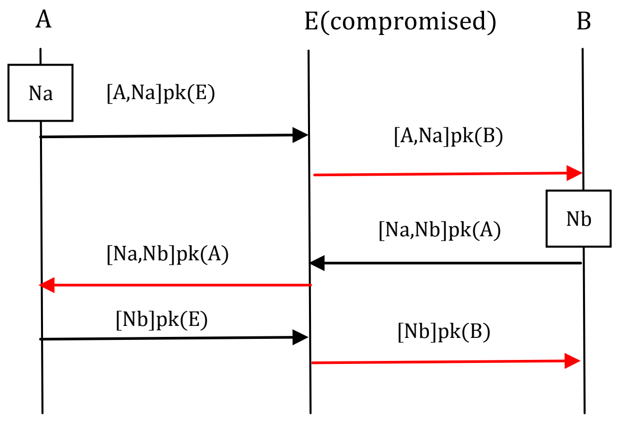

Figure B.1 — Man-in-the-middle-attack (G. Lowe 1995)

Secrecy defined as it stands can be referred as weak secrecy, since it does not care about partial disclosure of the message content. There are also probabilistic secrecy, indistinguishability, etc. But they are out of scope here.

B.3.2 Authentication

There is no general consensus on the meaning of authentication. Indeed, there is a hierarchy of authentication properties as it is shown by Lowe: aliveness, recentness, (non-)injective agreement, (non-)injective synchronization, etc. G. Lowe described, e.g., the *non-injective agreement* as follows:

> Initiator i is in agreement with responder r, whenever i as initiator completes a run of the protocol with r, then r as responder has been running the protocol with i. Moreover, i and r agree on all data variables.

Injective agreement insists furthermore that each run of i corresponds to a unique run of r.

A formal specification of authentication can, however, differ from a tool to another in a subtle way depending on the specification language and operational semantics. We give here an intuitive understanding of (non-)agreement as a trace property. First we need to introduce another event using a binary predicate **`Claim`**.

$$\texttt{Claim}(r, \mathbf{NI-Agree})\ \text{for the non} - \text{injective agreement holds for the role}\ r$$

Each trace of a protocol P can contain a number of instances of the claim event. In order that the non-injective agreement holds for the role r, the following should hold:

> In each trace where a claim event is executed at the ℓ-th transition step by a honest agent A who has communicated with honest agents only, each expected exchange of a message up to the point of the claim in the protocol should have been executed by expected honest agents before the ℓ-th transition step in the same trace.

More informally said, the non-injective agreement holds for r when

> the instantiated claim event (A, **`NI-Agree`**) is executed by an honest agent A at the ℓ-th transition of a trace, and when the trace is supposed to be created by honest agents only, then all exchanged messages, between two honest agents A, A'' who are supposed to have performed the sending and receiving events according to the protocol before the claim event, should have been sent or received by A, A''before the ℓ-th step in the same trace.

In case of the Needham-Schroeder public-key protocol

$$\begin{array}{l} \mathtt{Send}(I, R, \{N_i, I\}_{\mathrm{pk}(R)}); \\ \mathtt{Receive}(R, I, \{N_i, x\}_{\mathrm{pk}(I)}); \\ \mathtt{Send}(I, R, \{x\}_{\mathrm{pk}(R)}) \\ \mathtt{Claim}(I, \mathbf{NI} - \mathbf{Agree}) \end{array} \quad \text{and} \quad \begin{array}{l} \mathtt{Receive}(I, R, \{y, I\}_{\mathrm{pk}(R)}); \\ \mathtt{Send}(R, I, \{y, N_r\}_{\mathrm{pk}(I)}); \\ \mathtt{Receive}(I, R, \{N_r\}_{\mathrm{pk}(R)}) \\ \mathtt{Claim}(R, \mathbf{NI} - \mathbf{Agree}) \end{array}$$

the man-in-the-middle-attack in Figure B.1 says that the non-injective agreement fails for the responder role: the honest agent B believes to have communicated with the honest agent A and the message $\{N_b\}_{\mathrm{pk}(B)}$ is supposed to be sent by the honest agent A, but it is faked by the compromised agent E.