INTERNATIONAL STANDARD

ISO/IEC 29128-1

Second edition
2023-03

Information security, cybersecurity and privacy protection — Verification of cryptographic protocols —

Part 1: Framework

Reference number
ISO/IEC 29128-1:2023(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 27, *Information security, cybersecurity and privacy protection*.

This second edition cancels and replaces the first edition (ISO/IEC 29128:2011), which has been technically revised.

The main changes are as follows:

— removal of informal and paper-and-pencil proofs;

— deprecation of PAL levels;

— streamlining of technical requirements and explanations;

— minor editorial changes to bring the document in line with the ISO/IEC Directives Part 2, 2021.

A list of all parts in the ISO/IEC 29128 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

Many cryptographic protocols have failed to achieve their stated security goals because they are complicated and difficult to design correctly in order to achieve the desired functional and security requirements. This inherent difficulty means that protocols need to be rigorously analysed in order to find errors in their design. The goal of this document is to standardize a method for analysing protocols by proposing a clearly defined verification framework based on well-founded scientific methods.

This document proposes a standardization procedure analogous to what exists for cryptographic algorithms. National and international bodies have evaluation processes that instil a high degree of confidence that a standardized cryptographic algorithm meets the specific security requirements it was designed for. A similar process for cryptographic protocols would provide confidence that a verified protocol meets its stated security properties and can be used in security-critical systems.

The proposed verification process is based on state-of-the-art protocol modelling techniques using rigorous logic, mathematics, and computer science. It is designed to provide objective evidence that a protocol satisfies its stated security goals. Verification is not a guarantee of security; as with any modelling, the results are constrained by the scope and quality of the model and tools used.

Information security, cybersecurity and privacy protection — Verification of cryptographic protocols —

Part 1: Framework

1 Scope

This document establishes a framework for the verification of cryptographic protocol specifications according to academic and industry best practices.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

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

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
adversarial model
capabilities an *adversary* (3.2) has when attempting to attack a *cryptographic protocol* (3.4)

3.2
adversary
party attempting to disrupt the secure operation of a *cryptographic protocol* (3.4), with abilities defined by the *adversarial model* (3.1)

3.3
automated prover
tool used for evaluating the *security properties* (3.9) of a *cryptographic protocol model* (3.5)

3.4
cryptographic protocol
communication protocol which uses cryptography to perform security-related functions

3.5
cryptographic protocol model
formal cryptographic protocol specification (3.8) combined with an *adversarial model* (3.1) and a set of *security properties* (3.9)

3.6
cryptographic protocol specification
human-readable document which defines the functionality of a *cryptographic protocol* (3.4)

3.7
evaluator
party in the verification process defined by this document who evaluates a submitted protocol

3.8
formal cryptographic protocol specification
model of a *cryptographic protocol specification* (3.6) written in a machine-readable language

3.9
security property
security goal which a *cryptographic protocol* (3.4) is designed to guarantee, transcribed into a machine-readable language

3.10
soundness
property of a mathematical system in which every statement that can be proved is true

3.11
self-assessment evidence
proofs of *security properties* (3.9) produced by an automated prover run on a *cryptographic protocol model* (3.5)

3.12
submitter
party in the verification process defined by this document who submits a protocol for evaluation

4 Formal verification of cryptographic protocols

4.1 Methods for modelling cryptographic protocols

The goal of the formal verification of a cryptographic protocol is to obtain a formal proof that the protocol, as defined in a protocol specification, meets its security properties and objectives within a specified adversarial framework. Achieving this goal requires the construction of a protocol model. There are two common paradigms for defining a protocol model - the symbolic model and the computational model.[11] The standardization process in this document considers only the symbolic model, however both models are reviewed here for completeness.

In the symbolic model, aspects of cryptography are abstracted out of the model, so cryptographic primitives are represented as black box functions and cryptography is assumed to be perfect. The adversary is restricted to computations and algebraic operations that are part of the given primitives, such as encrypting a message using a key, verifying a signature on a message, or computing the Diffie-Hellman public value from a known private value. In particular, this means that if an adversary knows some encrypted text, the only way they can learn the plain text from it is if they also possess the relevant key. Abstracting away from purely cryptographic attacks allows for the construction of a simple protocol model which is focused on the security of the protocol rather than the underlying cryptography.

In the computational model, the cryptographic aspects are an integral part of the model. The method of showing security in the protocol is to prove that attacking the protocol is at least as difficult as breaking one of the hard problems that the cryptographic primitives are based on. The adversary may do any computations on values they know, provided the computations run in polynomial time.

Automated tools for the computational model are limited and much less mature than for the symbolic model. As such, proofs in the computational model are generally hand-written rather than automated. Once tooling for computational models evolves, verification of protocols using those proofs can be incorporated into this document.

Another paradigm for proving protocol security, known as composition, involves proving statements about small parts of protocols and then combining those pieces together to prove statements about the whole protocol. Universal composability is one major type of composability that has recently been used to prove security properties of protocols that are used in practice. Automated provers for composability are still in their infancy, but as they mature, verification of protocols using those proofs can be incorporated into this document.

4.2 Verification requirements

4.2.1 Methods of verification

The state-of-the-art methodology for verifying the security properties of cryptographic protocols is through the use of tools called automated provers. An automated prover takes in a description of a protocol along with descriptions of security properties for that protocol. The prover then attempts to either prove that, under certain assumptions, each security property holds or finds a sequence of messages which allows an adversary to violate the security property. These inputs are part of a cryptographic protocol model, which is defined fully in 5.3.

An automated prover may take advantage of computational power to verify complex security properties by checking many cases and sub-cases without human intervention. It also produces repeatable results which can be reviewed and verified by other parties. Automated provers require the protocol specification to be written in a language that the tool is able to parse; in this document, this is termed a formal specification.

4.2.2 Verification tools

Many automated provers currently exist for verifying security properties. In the future, new tools will surely be developed, and it is always possible for errors and bugs to be found in tools. As such, this document does not specify a list of eligible tools which can be used and instead specifies the properties that a tool shall have.

The only tools which are eligible to be used for this standardization process are automated provers, including model checkers, which are capable of accepting as input a cryptographic protocol model as described in 4.3.

Automated provers are based on an underlying mathematical framework, which is the foundation on which the proofs produced by the prover are based. In order to have confidence in the proof results, the soundness of the framework shall be verified. Many provers have papers claiming to prove soundness, which provide an excellent starting point for this verification.

Automated provers are software, and like all software the possibility of errors in the code exists. The tool shall be auditable, such that anybody is able to review the code for the tool and its dependencies.

Lastly, the tool shall produce results which are repeatable. This means that anyone possessing the inputs and a copy of the tool may reconstruct the results.

4.2.3 Bounded vs unbounded verification

Proofs taking advantage of automated tools can provide a particularly effective way to simplify the formal verification process. Automatic provers obtain predictable results since their soundness and completeness are already proven. Another important advantage of an automatic prover is the fact that they can use available computational power to solve particularly complex security properties; properties that would be out of reach of manual verification.

Tool-aided proofs require the specification of the cryptographic protocol to be written in a language that the tool can execute; in this document this is termed a formal cryptographic protocol specification.

Two types of verification tools are recognized by this document: model checkers and theorem provers. Confidence in a tool is not determined by its type but by whether it can handle unbounded sessions or not.

Finally, verification tools can be fully automated (automatic) or require guidance from the developer (semi-automatic).

4.3 Cryptographic protocol model

4.3.1 Description of a model

In order to create formal proofs of security properties, the construction of a cryptographic protocol model is required. For the purposes of this document, such a model consists of:

— a formal cryptographic protocol specification based on the protocol specification;

— an adversarial model defining the adversary's capabilities;

— a model of the desired security properties.

Annex A provides an example of a cryptographic protocol model.

Verification techniques are applied to the protocol model in an attempt to prove the correctness of the security properties. For each desired security property, these techniques can result in a proof of correctness for that security property, an attack that shows the protocol does not satisfy the desired security property, or they can fail to find either a proof or an attack. Such proofs of correctness of security properties will be referred to as self-assessment evidence.

4.3.2 Formal specification

Cryptographic protocol specifications are written in a way that humans can read and implement them. In order to use such a specification for modelling, it shall be re-written in a computer-readable language which can be interpreted by an automated prover. This new specification is called a formal cryptographic protocol specification and shall encapsulate all relevant aspects of the protocol.

The formal specification shall model all the messages which can be sent by the parties involved in the protocol. Typically, this is done by modelling the set of messages for each party as a separate role in the protocol, or by modelling messages individually.

The formal specification shall model all of the functions which are used in constructing the messages defined above. Common functions are often built into the automated prover, and any which are not shall be included in the formal specification. These functions include cryptographic functions such as encryption, signing and hashing, as well as non-cryptographic functions such as concatenation. If the modelled protocols make use of Diffie-Hellman, this should include, for example, functions for exponentiation and multiplication.

The formal specification shall model variables. Variables are used as inputs to and outputs of functions and also as components of messages sent over the network.

The formal specification shall contain an algebraic structure which defines the mathematical rules governing functions and variables. The structure defines the behaviour about how functions and variables interact with each other, in order to model the way cryptographic operations interact with each other.

EXAMPLE The functions for encrypting and decrypting a message m with a key k can be written as

$\mathrm{enc}(m,k)$ and $\mathrm{dec}(m,k)$

with the relationship

$\mathrm{dec}(\mathrm{enc}(m,k),k) = m$

Typically, most or all of the algebraic structure is built into the automated prover. Any algebraic relations which are needed for the protocol but not built into the prover shall be included as part of the model.

4.3.3 Adversarial model

4.3.3.1 General

The adversarial model constitutes the powers and abilities used by a hypothetical adversary to compromise the system. It defines what information the adversary has access to and what actions they are capable of taking. The starting point for most symbolic adversarial models is the Dolev-Yao model, and using this model is required for protocol verification in this document. As this is the most common model, it is also the model used by most automated provers. A brief description is provided in 4.3.3.3, and a more detailed explanation can be found in Annex E.

4.3.3.2 Network specification

The network specification explains the network operating environment of the protocol. Typically, this consists of a single public communication channel which is shared by all parties and which an adversary has full control over. However, additional channels where the adversary has less control are also part of some protocols and can be modelled by many tools.

4.3.3.3 Dolev-Yao model

The Dolev-Yao adversarial model defines three main types of abilities of an adversary.

The adversary has full control of the public portions of the network. They are able to read all messages sent over the network, send their own messages over the network and delete messages preventing them from being seen by others on the network. Messages sent over the network by the adversary will be of a type defined in the formal specification. Full network control is, in many cases, an unrealistically strong threat model, however all modern protocols are designed to withstand attack from adversaries with this capability.

The adversary can know and retain information. Any values they see on the network and any values they are able to compute can be stored indefinitely for later use.

The adversary can perform computations. The formal specification defines algebraic operations and functions which are part of the protocol being studied. An adversary may use all of these operations and functions with any of the information it has. This allows the adversary to learn new information derived from existing information.

EXAMPLE If an adversary is in possession of an encrypted message and the key that is needed to decrypt the message, then the adversary can learn the message.

Annex D provides more information on the Dolev-Yao model.

4.3.4 Submitting a model

If the automated prover does not have an embedded Dolev-Yao model, then a specification for the adversarial model shall be included as part of the protocol specification. If no adversarial model is submitted, it is assumed that the embedded Dolev-Yao adversarial model is being used.

It is possible to grant additional powers to an adversary to obtain a stronger proof of security. Granting additional powers is permitted in this document and such powers are defined as part of the adversarial model. Any such powers defined include a plain language description of the powers.

4.3.5 Security properties

Security properties define the security goals of a cryptographic protocol, such as authentication of a party or secrecy of a transmitted message. In the protocol model, security properties are statements about the formal specification, written in the same language, which the automated prover will attempt to verify.

4.3.6 Self-assessment evidence

After constructing the cryptographic protocol model, the model is run through the automated prover. The self-assessment evidence is the collection of proofs of security properties this process generates.

5 Verification process

5.1 General

This clause defines the process that shall be undertaken to complete a verification of a candidate protocol. There are two separate parties involved in the process: the submitter and the evaluator. The submitter is the party who creates a cryptographic protocol model for the protocol they wish to verify. The submitter would likely be a protocol designer or someone who wishes to use a protocol, but anybody can be a submitter. The evaluator is the party that verifies the self-assessment evidence produced from the cryptographic protocol. The evaluator shall be someone with the technical expertise to conduct the verification process defined in 5.3 and shall be independent from the submitter.

5.2 Duties of the submitter

The submitter shall provide the following to the evaluator:

— the cryptographic protocol specification which defines the protocol to be evaluated;

— the cryptographic protocol model produced based on the specification (as defined in 4.3);

— the self-assessment evidence produced by the automated prover (as defined in 4.3.6);

— the name and version number of the automated prover used to produce the self-assessment evidence;

— a copy of, or access to, the automated prover, if it is not readily available to the evaluator;

— a list of the claimed security properties written in plain language;

— a list of additional powers granted to the adversary written in plain language, if applicable.

Annex B provides an example of a protocol submission for evaluation.

5.3 Duties of the evaluator

5.3.1 Main duties

The evaluator is responsible for evaluating all material provided by the submitter. This consists of:

— evaluating the automated prover used to produce the self-assessment evidence;

— evaluating the cryptographic protocol model created by the submitter;

— evaluating the self-assessment evidence provided by the submitter.

5.3.2 Evaluating the prover

The evaluator shall evaluate the suitability of the prover being used. The evaluator shall examine the prover and verify that it meets all of the criteria defined in 4.2.2.

5.3.3 Evaluating the model

The evaluator shall verify that all aspects of the submitted model as defined in 4.3.2 are correct.

The evaluator shall ensure that the formal specification accurately models the protocol specification. Moreover, they shall ensure the model contains all relevant details from the protocol specification and that anything omitted from the model does not affect the security.

The evaluator shall ensure that any non-public communication channels used in the model are correctly implemented and that all messages are properly modelled on the correct channel.

The evaluator shall verify that the Dolev-Yao adversarial model is being used. If the adversarial model is built into the prover, then that will be evaluated; otherwise the adversarial model will be submitted as part of the model and that will be evaluated. If additional powers are granted to the adversary, the evaluator will verify that the plain language descriptions match the formal descriptions.

The evaluator shall ensure that the security properties which are modelled correspond to those which have been claimed by the submitter.

5.3.4 Evaluating the evidence

The evaluator shall evaluate the self-assessment evidence to ensure that the security properties have been proven. First, the evaluator shall run the cryptographic protocol model through the automated prover, verifying that the submitted self-assessment evidence is correct and repeatable. Then the evaluator shall interpret the self-assessment evidence and for each security property verify that the property has been proven.

5.3.5 Example evaluation

Annex C provides an example of the work performed by an evaluator when evaluating a submission.

Annex A
(informative)

The Needham-Schroeder-Lowe public key protocol

A.1 Protocol specification

The Needham-Schroeder-Lowe public key protocol is designed to provide mutual authentication between two parties known as the initiator and responder. The protocol is based on the Needham-Schroeder public key protocol, which was found by Lowe to be insecure. At the end of a successful protocol execution, the responder is guaranteed that they were communicating with the initiator with identity I, while the initiator is guaranteed that they have been communicating with the responder with identity R. In addition, the protocol guarantees the confidentiality of the values exchanged.

The protocol assumes that each party possesses the public key of the other party. The following three messages comprise an execution of the protocol:

a) the initiator generates a random value (nonce) N_I, and sends the message (I,N_I) encrypted using the public key corresponding to R;

b) the responder generates a nonce N_R, and sends the message $(N_\mathrm{I}, N_\mathrm{R}, R)$ encrypted using the public key corresponding to I;

c) the initiator sends the message (N_R) encrypted using the public key corresponding to R.

A.2 Protocol model

An example cryptographic protocol model for the Needham-Schroeder-Lowe public key protocol written for the Tamarin prover[33] is available at: https://github.com/tamarin-prover/tamarin-prover/blob/1.4.1/examples/classic/NSLPK3.spthy.

Annex B
(informative)

Example submission

B.1 General

In this annex, an example is provided of the items a submitter includes in their submission, as defined in 5.2. For this example, the Needham-Schroeder-Lowe protocol, as defined in Annex A is used.

B.2 Protocol specification

The protocol specification is given in A.1.

B.3 Protocol model

The protocol model is given in A.2.

B.4 Self-assessment evidence

The submitter includes the Tamarin output from their model. The output has been omitted from this example.

B.5 Automated prover

The model was created for and executed using the Tamarin prover version 1.4.1. The source code is available at https://github.com/tamarin-prover/tamarin-prover/releases/tag/1.4.1.

B.6 Security properties

a) The exchanged nonce values N_I and N_R are secret, known only to the initiator and responder.

b) The initiator and responder have injective agreement on N_I and N_R.

Annex C
(informative)

Example evaluation

C.1 General

In this annex, an example is provided of the steps performed by an evaluator in order to verify a submission. This example is based on the submission of the Needham-Schroeder-Lower protocol in Annex B.

C.2 Evaluation

C.2.1 Evaluating the prover

The tool used (Tamarin prover version 1.4.1) is an automated prover as required in 4.2.2. The underlying framework for Tamarin was evaluated and it has the required soundness property. The code for Tamarin was reviewed – it is well written according to standard coding best practice and no errors affecting the soundness were found. Tamarin produces results which are repeatable.

C.2.2 Evaluating the model

The provided protocol specification and model were reviewed, and the model was found to accurately represent the protocol. Tamarin uses a Dolev-Yao adversarial model. The security properties modelled accurately correspond to the descriptions provided in the submission.

C.2.3 Evaluating the evidence

The provided model was run using the Tamarin prover version 1.4.1. The output produced matched the output provided in the submission. The evidence shows that the security properties listed in the submission hold.

Annex D
(informative)

Dolev-Yao model

In the Dolev–Yao model, there are two kinds of agents: honest agents and the adversary. What distinguishes an honest agent from the adversary, is that the former should execute the protocol as specified whereas the adversary has much more freedom. In fact, the adversary has complete control over the network, this means that the adversary can:

— read, redirect, insert, delete and modify messages on all public channels;

— store information intercepted on a public channel;

— craft and send new messages.

Messages in this model can include any type of information including identifiers and cryptographic keys. Compound messages can be formed by applying cryptographic primitives to messages or by pairing messages. The adversary can apply cryptographic primitives to messages already known as well as randomly generated values. Since the Dolev-Yao model assumes perfect cryptography, the adversary cannot successfully decrypt an encrypted message without knowing the key. The adversary can however generate their own values such as keys and nonces, and use them in cryptographic operations.

As a side-effect, the adversary can compromise any number of agents and maybe even learn their private information. The adversary also has some initial knowledge which can include all publicly available values, such as plain text identities and the public keys of honest agents. Any unencrypted values such as nonces are also available to the agent. From there, the adversary can use any available cryptographic primitive to generate new values.

Typically, a list of communication channels is provided, each of which with its own properties. For instance, one can specify a single public communication channel which is under complete control of the attacker. But it can be refined further, distinguishing a more 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.