ISO/IEC JTC 1/SC 22/WG 23 N 0454

Date: 2013-06-03

ISO/IEC IS 17960

Secretariat: ANSI

Information Technology—Programming languages, their environments and system software interfaces—Code Signing for Source Code

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International standard
Document subtype: if applicable
Document stage: (20) development stage
Document language: E

Copyright notice

This ISO document is a working draft or committee draft and is copyright-protected by ISO. While the reproduction of working drafts or committee drafts in any form for use by participants in the ISO standards development process is permitted without prior permission from ISO, neither this document nor any extract from it may be reproduced, stored or transmitted in any form for any other purpose without prior written permission from ISO.

Requests for permission to reproduce this document for the purpose of selling it should be addressed as shown below or to ISO's member body in the country of the requester:

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

Reproduction for sales purposes may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Table of Contents

Foreword v
Introduction vi
1. Scope 1
2. Conformance 1
3. Normative references 1
4. Terms and definitions 2
5. Concepts 3
6. Requirements 5
6.1. Certificates 5
6.2. Hash-Code 5
6.3. Initial Code Signing 6
6.4. Modifying Previously Signed Code 6
6.5. Revision Format 6
Annex A (*informative*) Notional Code Signing Process 7
Bibliography 9

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. 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 IS 17960, which is an International Standard, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

Introduction

Source code is written and is used in many critical applications. Knowing that the source code being relied upon is the same as that which was used in testing is vital to ensuring the safety and security of a particular application. Given the ease with which source code can be modified, some method of protecting the integrity of the source code is necessary. Sequestration of the source code is one method, but ensuring protection in that way is impractical and unreliable. Virtual protection through the use of a digital signature offers a practical solution and provides integrity even though the source code may traverse an insecure supply chain.

Modifications to source code are frequently made to correct the software or to adapt it for other purposes. Rarely are the modifications made by the original author. Revision control software facilitates tracking of the software changes, but such tracking can easily be spoofed. Digital code signing provides a means to restrict the ability to spoof by assigning a responsible party to each of the modifications as they are made.

This International Standard specifies the necessary metadata for signing source code in a manner that allows signatures to be shared among applications to ensure the integrity and a means for reversing the application of the signatures to roll back the source code to previously signed versions. Annex A contains a step by step description of a typical application of source code signing. A bibliography lists documents that were referenced during preparation of this standard.

Information Technology — Programming Languages — Code Signing for Source Code

1. Scope

This International Standard specifies a language-neutral and environment-neutral description to define the methodology needed to support the signing of software source code. It is intended to be used by originators of software source code and the recipients of their signed source code. This standard is designed for transfers of source code among disparate entities.

The following areas are outside the scope of this specification:

- Determination of the trust level of a certification authority
- Format used to track revisions of source code files
- Digital signing of object or binary code
- System configuration and resource availability

2. Conformance

An implementation of code signing conforms to this International Standard if it meets the requirements specified in Clause 6.

Clause 5 provides an overview of the concepts of code signing. Annex A is informative and provides a possible scenario of usage for the standard specified in Clause 6.

3. Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 10118–3:2004, *"Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash functions"*

ISO/IEC 13888-1:2009, "*Information technology — Security techniques — Non-repudiation – Part 1: General*"

ISO/IEC 9594-8:2008, "Information technology — Open Systems Interconnection — The Directory: Public-key and attribute certificate frameworks"

ITU-T Recommendation X.509: 2005, "*Information Technology —Open Systems Interconnection — The Directory: Public-Key and attribute certificate frameworks*"

4. Terms and definitions

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

4.1

certificate

entity's data rendered unforgeable with the private or secret key of a certification authority [9]

4.2

certification authority

authority trusted by one or more users to create and assign certificates [9]

4.3

changeset

set of all changes that are applied to a configuration to derive a new configuration

4.4

digital signature

data appended to, or a cryptographic transformation of, a data unit that allows the recipient of the data unit to prove the source and integrity of the data unit and protect against forgery e.g. by the recipient [9]

4.5

hash-code

string of bits that is the output of a hash-function [9]

4.6

hash-function

function which maps strings of bits to fixed-length strings of bits, satisfying the following two properties:

- it is computationally infeasible to find for a given output an input which maps to this output;
- it is computationally infeasible to find for a given input a second input which maps to the same output [9]

4.7

originator

entity that sends a message to the recipient or makes available a message for which non-repudiation services are to be provided [9]

4.8

private key

key of an entity's asymmetric key pair which should only be used by that entity [9]

4.9

public key

key of an entity's asymmetric key pair which can be made public [9]

4.10

public key certificate

public key information of an entity signed by the certification authority and thereby rendered unforgeable [9]

4.11

recipient

entity that gets (receives or fetches) a message for which non-repudiation services are to be provided [9]

4.12

snapshot

verbatim copy of a configuration

ConformanceAn implementation of code signing conforms to this International Standard if it meets the requirements specified in ClauseClause 5 is informative, providing an overview of the concepts of code signing. Annex A, also informative, provides a possible scenario of usage for the standard specified in Clause 6.

5. Concepts

This clause is informative.

Code signing is a technique for providing a digital signature for source code to support a verification of the origin and a verification that the code has not been altered since it was signed.

Code signing can provide several valuable functions such as:

- knowledge of the origin of the source code
- confidence that the source code has not been accidentally or maliciously altered
- verification of the identity of the responsible party for the source code

- accountability for the source code
- non-repudiation of the source of the source codenon-repudiation of the source of the source code
-

Code Signing identifies to customers the responsible party for the source code and confirms that it has not been modified since the signature was applied. Verification of the origin of the source of the software is extremely important since the security and integrity of the receiving systems can be compromised by faulty or malicious code. In addition to protecting the security and integrity of the software, code signing provides authentication of the author, originator or distributor of the source code, and protects the brand and the intellectual property of the developer of the software by making applications uniquely identifiable and more difficult to falsify or alter maliciously.

When source code is associated with an originator's unique signature, distributing source code on the Internet is no longer an anonymous activity. Digital signatures ensure accountability, just as a manufacturer's brand name ensures accountability with packaged software. Distributions on the Internet lack this accountability and code signing provides a means to offer the needed accountability. Accountability can be a strong deterrent to the distribution of harmful code. Even though software may be acquired or distributed from an untrusted site or a site that is unfamiliar, the fact that it is signed by a known and trusted entity allows the software to be used with confidence that it has not been changed as compared to the most recently signed version.
In addition to the valuable functions that code signing offers, this International Standard will specifically facilitate the following capabilities:

- a tracking mechanism to show what has been altered in the source code and by whom
- multiple signatures to allow for an audit trail of the signed source code
- versioning information
- storage of other metadata about the source code

The capability for a tracking mechanism and multiple signatures for one piece of source code is needed in some cases in order to create a digital trail through the origins of the source code. Consider a signed piece of source code. Someone should be able to modify a portion of the source code, even if just one line or even one character, without assuming responsibility for the remainder of the source code. A recipient of the source code should be able to identify the responsible party for each portion of the source code. For instance, a very trustworthy company A produces source code for a driver. Company B modifies company A’s source code for a particular use. Company B is not as trusted or has an unknown reputation. The recipient should be able to determine exactly what part of the source code originated with company A and what was added or altered by company B so as to be able to concentrate their evaluation on the sections of source code that company B either added or altered. This necessitates a means to keep track of the modifications made from one signed version to the next.

An alternative scenario is source code offered by company B that contains source code from company A. Company B does not alter company A’s source code, but incorporates it into a package or suite of software. It would be useful to a customer to be able to identify the origin of each portion of Company B’s software package.

6. Requirements

The code signing standard described below is intended to be language and platform independent. To assist in understanding code signing, Annex A provides an overview of the code signing process from a conceptual perspective.

Throughout this standard, *originator* refers to the person or organization that is signing the source code. *Recipient* refers to the person or organization that is receiving the signed source code.

6.1. Certificates

The originator shall obtain an X.509-compliant certificate. The level of trust in the Certification Authority (CA) that issues the X.509-compliant certificate is an important factor in the amount of trust associated with the signed code. The CA should be a trusted party to both the originator and potential recipients. Though very important to the execution of a trusted transfer of software from an originator to a recipient, the establishment or determination of the trust level associated with a certification authority is beyond the scope of this standard.

Protection of the originator’s private key shall be ensured to prevent impersonation by others. The private key part of the originator’s certificate shall not be compromised from the control of whoever is authorized to sign the code.

6.2. Hash-Code

A hash-code shall be generated by applying a hash-function to the source code. A hash-code is used since public key signing of large source code files is inefficient. The default hash-function shall be the Secure Hash Algorithm-256 (SHA-256). Alternatively, hash-functions that are specified in 10118-3:2004 or its later revisions could be used.

The hash-code shall be encrypted with the private key of the originator to generate a digital signature for the source code file. The recipient shall then use the originator’s public key to verify that the source code file has not been altered since it was digitally signed so that the origins of the source can be traced back to the first signed version.

6.3. Initial Code Signing

The initial signing of a source code file shall be in any format such as snapshot or changeset. If changeset is used, it should be based on an empty file.

6.4. Modifying Previously Signed Code

Sufficient information shall be recorded in a signed version to allow the source code file and digital signature of the previously signed version to be recovered. This allows the series of modifications from one version to the next, which can be thought of as encapsulations, to be reversed one at a time.

The information contained in each encapsulation shall contain sufficient revision control information in order to recreate the previous version. Once an encapsulation is reversed, the recipient shall be able to use the digital signature of the encapsulated version to verify its integrity.
A mechanism shall allow for the recreation of the most recently signed version of the source code and a record of all changes that distinguish any signed version from any preceding signed version. It is implementation-defined whether intermediate unsigned versions can also be recreated by this mechanism.

6.5. Revision Format

This International Standard is not prescriptive as to which format shall be used to create or track revisions. A conforming implementation of this International Standard shall provide specifications so that recipients can reconstitute the previously signed version.

Annex A

(*informative*)

Notional Code Signing Process

This annex describes the series of steps in a typical implementation of code signing of source code.

1. **The originator obtains an X.509-compliant certificate from a global certification authority**

2. **The originator develops source code or modifies previously signed source code**

3. **The originator uses a hash-function to calculate a hash-code of the source code file**

    1. If the code has not been previously signed, the source code file is designated as the baseline version.

    2. If the code has been previously signed, sufficient information is documented and available to the recipient to allow the changes to be undone to revert to any of the previous versions created since the initial baseline version, though versions must be undone in the reverse order that they were signed.

    3. For either case a one-way hash-code of the source code is calculated using a hash-function.

4. **The originator uses his or her private key to encrypt the hash-code to produce a digital signature of the source code file**

5. **The source code file and its associated digital signature are transmitted to the recipient**

6. **The recipient obtains a trusted copy of the public key of the originator. This can be achieved by the recipient obtaining a copy of the public key certificate of the originator, and verifying it using a trusted copy of the public key of the CA that generated the certificate.**

7. **The recipient uses the same hash-function as the originator to produce a hash-code of the source code file**

8. **The recipient decrypts the digital signature using the originator's public key to reveal the hash-code**

9. **The recipient compares the two hash-codes**

If the signed hash-code provided by the originator matches the recipient's hash-code, the source code file has not been altered since it was digitally signed.

To verify previously signed versions of the source code, the version signed most recently to the current one is “unwrapped” from the current version. This allows a reconstruction of each previously signed version in sequential order.

Bibliography

[1] *Code-Signing Best Practices*, http://msdn.microsoft.com/en–us/windows/hardware/gg487309.aspx, July 25, 2007

[2] *How Code Signing Works,* https://www.verisign.com/code-signing/information-center/how-code-signing-works/index.html, 2011

[3] *Introduction to Code Signing*, http://msdn.microsoft.com/en-us/library/ms537361(VS.85).aspx, June 21, 2011

[4] *ISO/IEC 9594-8:2008, "Information technology — Open Systems Interconnection — The Directory: Public-key and attribute certificate frameworks"*

[5] ISO/IEC 9796-2:2010, Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms

[6] ISO/IEC 9796-3:2006, Information technology – Security techniques – Digital signature schemes giving message recovery – Part 3: Discrete logarithm based mechanisms

[7] ISO/IEC 9899:2011, Information technology – Programming languages – C

[8] ISO/IEC 10118-3:2004, Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions

[9] ISO/IEC 13888-1:2009, Information technology – Security techniques – Non-repudiation – Part 1: General

[10] ISO/IEC 14750:1999, Information technology – Open Distributed Processing – Interface Definition Language

[11] ISO/IEC 14888-1:2008, Information technology – Security techniques – Digital signatures with appendix – Part 1: General

[12] ISO/IEC 14888-2:2008, Information technology – Security techniques – Digital signatures with appendix – Part 2: Integer factorization based mechanisms

[13] ISO/IEC 14888-3:2006, Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms

[14] ITU-T Recommendation X.509:2008, Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, http://www.itu.int/rec/T-REC-X.509/en

[15] ITU-T Recommendation X.509:2005, Information Technology - Open Systems Interconnection - The Directory: Public-Key and Attribute Certificate Frameworks, http://www.itu.int/rec/T-REC-X.509/en

[16] Steve Mansfield-Devine, *A Matter of Trust*, Network Security, Volume 2009, Issue 6, June 2009