
Testing And Signing With Symbian Platform Security

Version 1.1
October 17, 2005

S
Y
M
B
I
A
N
O
S

Legal Notice

Copyright © 2005 Nokia Corporation. All rights reserved.

Nokia and Nokia Connecting People are registered trademarks of Nokia Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

Disclaimer

The information in this document is provided “as is,” with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Furthermore, information provided in this document is preliminary, and may be changed substantially prior to final release. This document is provided for informational purposes only.

Nokia Corporation disclaims all liability, including liability for infringement of any proprietary rights, relating to implementation of information presented in this document. Nokia Corporation does not warrant or represent that such use will not infringe such rights.

Nokia Corporation retains the right to make changes to this specification at any time, without notice.

License

A license is hereby granted to download and print a copy of this specification for personal use only. No other license to any other intellectual property rights is granted herein.

Contents

1.	Introduction	5
2.	The Basics of Signing.....	5
2.1	An Example of Signing.....	5
2.2	Obtaining the Trust.....	6
2.2.1	The source of the application	6
2.2.2	The contents of the SIS file	6
2.2.3	In practice	6
3.	Symbian Security Model.....	7
3.1	Reasoning Behind Symbian Platform Security Model	7
3.2	Symbian Platform Security Model in Practice.....	7
3.2.1	Trusted computing base.....	7
3.2.2	Data caging	7
3.2.3	Capabilities	8
3.3	Assigning Capabilities	8
4.	Symbian Security Model in Application Development	11
4.1	Defining the Application	11
4.2	Testing the Application.....	11
4.2.1	Symbian Developer Certificate.....	11
5.	Terms and Abbreviations	13
6.	Evaluate This Resource.....	13

Change History

September 5, 2005	Version 1.0	Initial document release
October 17, 2005	Version 1.1	Information on the Developer Certificate pricing updated

1. Introduction

This document describes first the basics of signing, and then widens the scope to the Symbian Security Model. It is very important that the reader understands the basics. Otherwise the Symbian Security Model may be difficult to comprehend.

2. The Basics of Signing

Signing is based on a standard public key infrastructure (PKI) model. One way to explain the PKI model without interfering with the technical details is that it is trust between two parties.

2.1 An Example of Signing

As an example, the lock on a door can be opened with several keys. There is trust between the lock and the key that opens the lock. The door will not open with a wrong key without using force. A crowbar can also be used to open the lock. However, in this case force is used and it cannot be considered a trusted way to open the lock. In some places opening the door in such a way may trigger an alarm. To summarize, trust exists between the lock and the right keys, but not between the lock and the crowbar.

Now the question is, where does the trust come from? In the lock example, buying a lock from an authorized locksmith is the most secure way. When you obtain the keys, you can be sure that there are no duplicates. You can get a crowbar from the local hardware store and so does anyone else. Thus, trust is not available to crowbar users trying to open the door.

Signing is implemented in the same way. There needs to be an element of trust (the keys for the lock) that uses the trust on a desired target (the lock) to get the desired result (opening of the door). If you use some other means (the crowbar) to access the target, a warning will be displayed (the alarm goes off).

When this model is applied to the Symbian installation process, the content is installed to the device (door) in a SIS file (that wants to enter through the door) using the device's installer (lock) application. The trust is then measured between the installer and the SIS file. For this purpose, the device has an entity called root certificate (lock mechanism) and the trust entity in the SIS file is a digital signature (the key or the crowbar).

When the SIS file is introduced to the installer application, the installer application checks the SIS file to see if it has been digitally signed (what type of keys it uses to open the lock). If the SIS file does not have a signature, the crowbar is used. The door is opened and the application is installed, but a warning message is displayed to the user.

If the SIS file has been digitally signed, the installer will check what type of signature it has. After checking the signature, it will compare the signature with all the phone's certificates that are used for the application installation. If none of the certificates trust the signature, the installation will have the same result as the crowbar choice. The application is installed but a warning message is displayed to the user. However, in this case the user can see which signature the SIS file has; in other words, who has signed the application.

If the SIS file has a signature that is trusted by one of the certificates used by the application installer, no warning message is displayed and the application is installed. In this case, the user can see who has signed the application.

2.2 Obtaining the Trust

The process where the application is signed needs to be very secure, in order for the users to trust the signature and the certificate in application installation. This means that only certain trusted parties can sign the applications. A party issuing certificates and signatures is called Certificate Authority (CA).

Before a SIS file can be signed, certain issues need to be checked. This naturally depends on the nature of the signature, but in the current models there are two checks:

1. The source of the application.
2. The content of the SIS file against a certain criteria.

2.2.1 The source of the application

The source of the application can be verified by issuing an identity certificate. This means that the provider of the application needs to give the CA a certain level of proof of their existence. After the proof has been verified, the CA can issue the application provider an identity certificate. The provider can then sign the SIS file with that identity certificate to certify:

1. The source of the application.
2. The integrity of the SIS file.

The signature protects the integrity of the SIS file. If a signed SIS file is changed, the signature is broken and the integrity of the SIS file is compromised.

Signing the SIS file does not offer any trust to the application installation process because the identity certificate is only for certifying the identity of the SIS file provider.

2.2.2 The contents of the SIS file

Because the SIS files can contain basically anything targeted to Symbian OS phones, the content of the SIS file can be checked with various methods. The most straightforward way is to install the contents of the SIS file to the targeted device and see what happens. Especially in cases where the content is an application with functionalities, a criterion to test that is needed. For non-functional content, such as themes, a visual check should be adequate. This is usually done already by the provider, but it still needs to be checked that the content is really non-functional before it can be signed.

2.2.3 In practice

For Symbian applications there is one industry-wide application testing and signing program available. That is trusted by the industry to sign the applications in a way that users can install them with minimum risk. The program is called Symbian Signed (www.symbiansigned.com).

To support the program, phones need to have the root certificate used by Symbian Signed. Symbian OS device manufacturers should add that to their devices. The manufacturers supporting the program are directing the developers to Symbian Signed to get their applications signed before entering the market.

3. Symbian Security Model

3.1 Reasoning Behind Symbian Platform Security Model

Mobile devices are gaining more and more features and thus they are becoming more valuable to their users with the amount of important information they possess. However, as mobile devices they are not the same as computers, and the idea is to make sure users still find their phones easy to use, reliable, secure, and trustworthy. For this purpose, the Symbian Platform Security Model is introduced. The purpose of the model is not to close the phones but to enable users to continue using their mobile phones as before in an easy and trusted manner.

3.2 Symbian Platform Security Model in Practice

There are three main modules in the Symbian Security Model:

1. Trusted computing base.
2. Data caging.
3. Capabilities.

3.2.1 Trusted computing base

The trusted computing base is a collection of software that enforces capabilities and data caging. It contains the kernel, the file system, and the software installer. This is the controlling part of the operating system for the platform security model.

3.2.2 Data caging

Data caging means that the applications and the users have access only to certain areas of the file system. In practice the applications can access their own private directories and directories that are marked as open. It means, for example, that one application cannot access another application's private directory and data.

The access is restricted as follows:

`\resource`

Location for application icons, bitmaps, etc. Writing allowed only at application installation. Everyone can read the folder.

`\sys`

Location for binaries, including application installation registry and root certificates. Writing allowed only at application installation. Reading allowed to backup the application.

`\private`

This is a private playground for each application. Reading and writing is only allowed to the application's own directory. Backup software has read and write access to this directory.

all the rest

Access to all the other folders is free for all, for example, user's own photos, music, and documents.

3.2.3 Capabilities

A capability grants access to a set of APIs and files. These can be divided into three:

1. Open to all
 - APIs in this category enable development of all the basic applications, for example, most of the single player games.
2. Granted through Symbian Signed
 - Part of these capabilities are available after passing Symbian Signed testing.
 - More sensitive capabilities require declarative justification why the application needs access to such a capability.
3. Granted through the manufacturer
 - Access to all the resources of the phone.
 - Needed by only a few applications:
 - Target test applications
 - Anti-virus applications
 - File crypto

3.3 Assigning Capabilities

Capabilities required by the application are defined at the design phase of the application. The application binaries will include an MMP file, which includes information of the capabilities that the application uses. At the installation phase the installer compares the list of capabilities in the MMP file with the capabilities that are available through signing. If the root certificate is allowed to grant the required capabilities, the installation can continue.

The following illustration gives an idea of how the capabilities are divided between an application signed through Symbian Signed and an application signed by the manufacturer.

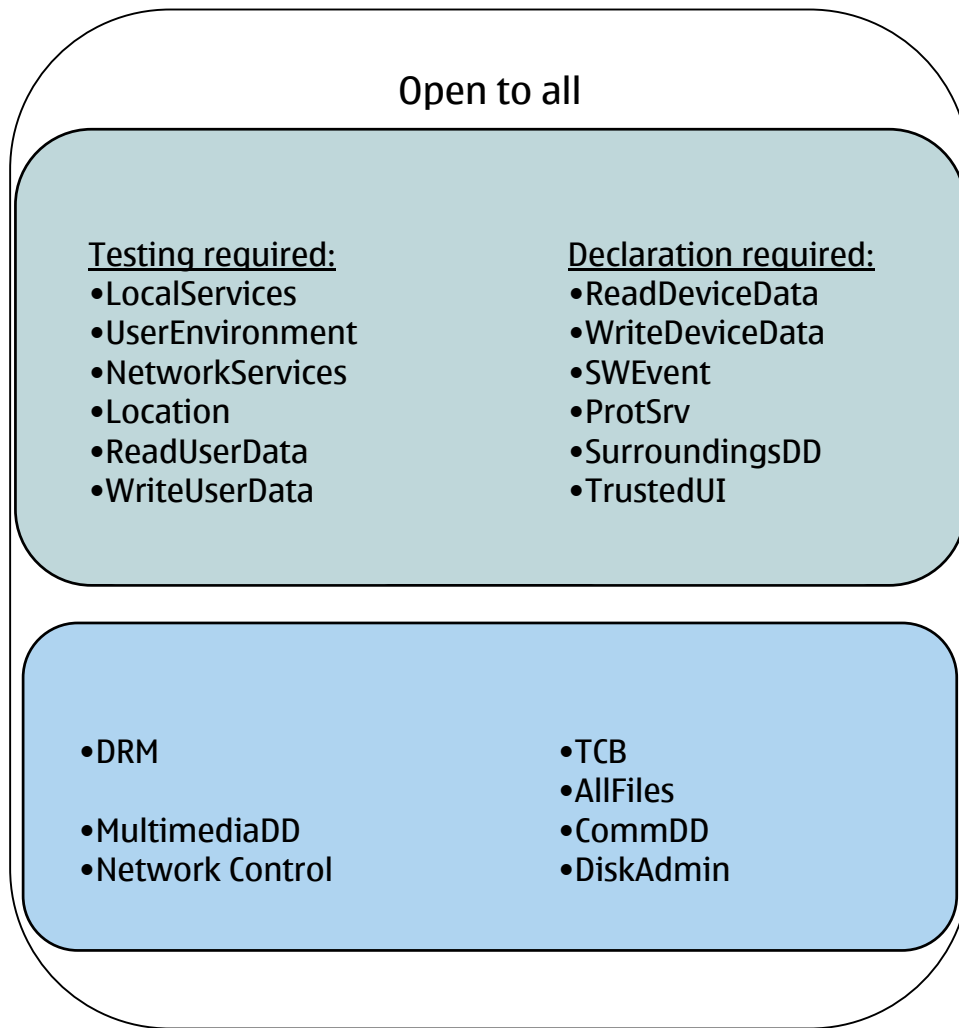


Figure 1: Comparison of capabilities

Table 1 provides more information on what the separate capabilities mean in practice.

	Capability	Description
1	NetworkServices	This capability is for, e.g., dialing a number or sending a text message.
2	LocalServices	This capability is for sending or receiving information through USB, IR, and point-to-point Bluetooth profiles.
3	ReadUserData	Grants read access to user data. System servers and application engines are free to grant this restriction level to their data.
4	WriteUserData	Grants write access to user data. Again, system servers and application engines are free to grant this restriction level to their data.
5	Location	Grants access to the location of the phone.

	Capability	Description
6	UserEnvironment	Grants access to live confidential information about the user and his/her immediate environment.
7	PowerMgmt	Grants the right to kill any process in the system or to switch machine state (turn the phone off).
8	MultimediaDD	Controls access to all multimedia device drivers (sound, camera, etc.).
9	ReadDeviceData	Grants read access to sensitive system data.
10	WriteDeviceData	Grants write access to sensitive system data.
11	DRM	Grants access to protected content.
12	TrustedUI	This capability differentiates "normal" applications from "trusted" applications. If a trusted application is displaying something on the screen, a normal application cannot fake it.
13	ProtServ	Grants the right to a server to register with a protected name. Protected names start with an "!" (exclamation point). The kernel will prevent servers without ProtServ capability from using such a name, and therefore will prevent protected servers from being impersonated.
14	NetworkControl	Grants the right to modify or access network protocol controls.
15	SwEvent	Grants the right to generate and capture software key and pen events.
16	SurroundingsDD	Grants access to logical device drivers that provide input information about the surroundings of the phone.
17	TBC	Grants access to /sys and /recourse directories in the phone.
18	CommDD	Grants access to communication device drivers.
19	DiskAdmin	Grants the right to disk administration functions, such as formatting a drive.
20	AllFiles	Grants visibility to all files in the system and extra write access to files under /private.

Table 1: Description of capabilities

As described earlier, some capabilities are granted by the device's manufacturer. The manufacturer will use its discretion before granting the capabilities. Usually strong enough business reasoning is sufficient to gain the capabilities.

For a developer to be able to get manufacturer capabilities for the application, it is necessary to contact the manufacturer in question for more details.

4. Symbian Security Model in Application Development

As can be understood from the security model description, it is important to know which capabilities are needed by the application and how to obtain the needed capabilities.

4.1 Defining the Application

At the phase where the application is on the drawing board, that is, being planned and defined, there are two important issues that need to be taken into consideration:

1. What do the Symbian Signed criteria require from the application?
2. Which capabilities, if any, does the application require?

4.2 Testing the Application

The application can be tested with the emulator of an SDK. It is advisable to also test the application on a real device in a live network — the emulators seldom get any incoming calls. If the application requires capabilities that require digital signature, testing must be done first with the emulator and secondly a Developer Access Certificate can be acquired through Symbian Signed for the application to obtain the desired capabilities for testing the application on a device.

4.2.1 Symbian Developer Certificate

The Symbian Developer Certificate can be used by the developer to sign their applications in order to obtain the restricted capabilities for device testing. The certificate is restricted to a certain set of IMEIs and the set cannot be changed. There are a few requirements to get a Symbian Developer Certificate. These are shown in Table 2.

Number of IMEIs	Authentication	Capabilities
1	Symbian Signed account	Local Services, User Environment, Network Services, Location, Read User Data, Write User Data, SWEvent, SurroundingsDD, ProtSrv, Power Mgmt
Up to 20	VeriSign ACS Publisher ID and Symbian Signed account	As above + ReadDeviceData & Write Device Data, Trusted UI
Custom	VeriSign ACS Publisher ID, Symbian Signed account and manufacturer support	As above + DRM, Network Control, MultimediaDD, TCB, All Files, CommDD, Disk Admin

Table 2: Requirements for the Symbian Developer Certificate

In general, the process goes as follows:

1. The developer enters the Symbian Signed portal and registers.
2. The developer uses a request tool to send a request for the developer certificate.
 - The VeriSign ACS Publisher ID may be needed at this phase. Note that the VeriSign ACS Publisher ID incorporates a yearly fee.
3. The certificate is created and sent back to the developer.
 - The certificate is valid for 6 months from the date the Developer Access Certificate was received.
 - The Symbian Developer Certificate is free of charge; however, the developer may need the VeriSign ACS Publisher ID, which is subject to a fee.

The process of obtaining the custom developer certificate is done through the same portal. However, there the manufacturer will need to approve the request before the developer can get a hold of the developer certificate.

5. Terms and Abbreviations

Term or abbreviation	Meaning
API	Application Programming Interface
CA	Certificate Authority
PKI	Public Key Infrastructure
SIS	Installer file format used in Symbian OS
VeriSign ACS Publisher ID	VeriSign's product, an identity certificate. "Authenticated Content Signing Publisher ID"

6. Evaluate This Resource

Please spare a moment to help us improve documentation quality and recognize the resources you find most valuable, by [rating this resource](#).