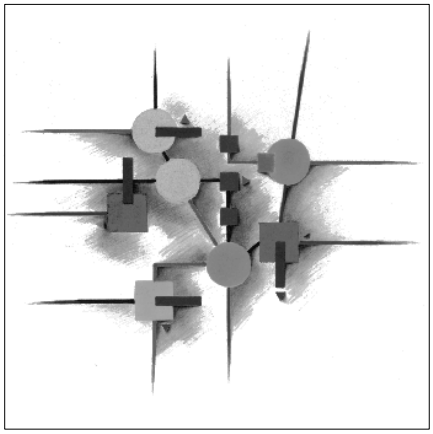
Secure Store & Forward (SSF)
Test Plan



SYSTEM R/3

Version 1.1

Release 4.5B

Copyright

1 Test Plan SSF (Secure Store & Forward) 4
1.1 Test Objective 4
1.2 Test Environment 5
1.3 Test Scenario Overview 5
2 The Test Procedure 6
2.1 Preparing the Environment 6
2.1.1 Application Server 6
2.1.2 Front end 6
2.1.3 Installation Functional Test 6
2.2 Installing the Security Product on the Application Server 6
2.3 Installing the Security Product on the Frontend Machines 6
2.4 Enabling SSF on the Frontend Machines 7
2.5 Enabling SSF on the R/3 Application Server 7
3 Using the Stand-Alone Test Tool SSFTEST 8
3.1 Installing SSFTEST 8
3.1.1 The Security Toolkit's SSF API Library 8
3.1.2 SSF Users 8
3.1.3 Specify Algorithms 9
3.1.4 Sample Tests 10
3.2 Executing SSFTEST on the Frontend Machines 10
3.3 Executing SSFTEST on the Application Server 11
3.4 Reading the Output Protocol of SSFTEST 11
3.5 Sample SSFTEST Output 12
4 R/3 Test Report SSFTEST 21
4.1 Testing SSF on the Frontend Machines 21
4.2 Testing SSF on the Application Server 22
4.3 Combined Front End and Application Server Test 24
4.4 Reading the Output Protocol of Report SSFTEST 25

1 Test Plan SSF (Secure Store & Forward)

With the Secure Store & Forward (SSF) functions, R/3 data and documents are “wrapped” in secure formats – the so-called “security wrapper” – before they are saved on data carriers or transmitted over (possibly) insecure communication links. A *digital signature* makes sure that the data is not falsified, that the sender (signatory) can be clearly determined, and that proof of award of contract exists. The subsequently assigned *digital envelope* makes sure that the contents of the data are only visible to the intended recipient(s). As a result, no security gaps arise, even if the data is temporarily stored during transport or at its destination. These two principle mechanisms use public-key technology.

To guarantee the interoperability of external security products with the SAP Software, the external product has to be certified for the BC-SSF interface by the SAP Integration and Certification Center (ICC). This document describes the certification tests that the security product has to pass to receive the BC-SSF certificate.

For general information about SSF, see the White Paper *Secure Network Communications and Secure Store and Forward Mechanisms within the SAP R/3 System* at http://www.sdn.sap.com/sdn/icc.sdn?page=secure_store_and_forward_and_digital_signatures.htm .

For complete information about the BC-SSF certification, see

http://www.sdn.sap.com/sdn/icc.sdn?page=secure_store_and_forward_and_digital_signatures.htm

1.1 Test Objective

The BC-SSF certification tests cover the security product's installation and configuration together with SAP Software client and server components.

After installation and configuration, the test tool analyzes the runtime behavior of the third-party security software by simulating the behavior of SAP Software components that use the SSF layer. Additionally, the stand-alone test tool SSFTEST collects statistical data about the runtime performance of individual SSF-API function calls and checks the interoperability with R/3 using sample data.

To further validate the interoperability with the SAP Software components, the cryptographic functions are called directly from R/3.

1.2 Test Environment

The BC-SSF certification procedure requires the following pre-configured hardware and software.

Provided by SAP:

- A Microsoft Windows NT 4.0 (or newer) Server running an R/3 application server Release 4.5B (or newer), connected to a R/3 System Release 4.5B (or newer).
- A Microsoft Windows NT 4.0 (or newer) Server or Workstation with a local SAP frontend installation for Win32, R/3 Release 4.5B (or newer).
- The SSF-API test tool SSFTEST used for the BC-SSF certification, precompiled for the hardware platforms mentioned above.

Provided by Partner:

- The third-party security software which supports the hardware platforms mentioned above and is to be examined for interoperability with SAP Software components through the BC-SSF interface.
- Any installation and configuration tools required as part of the security product's infrastructure.
- A preinstalled SAP R/3 Release 4.5B application server and system, as well as the frontend installation and security product installations on additional hardware other than those mentioned above (either bring to the SAP certification site or request an on-site certification at the Partner or Customer site).

1.3 Test Scenario Overview

The interoperability of the third-party security software and SAP Software components over the BC-SSF-Interface are tested using the following steps and scenarios:

- Installing and configuring the third-party security software on the SAP R/3 application server.
- Installing and configuring the third-party security software on the SAP frontend computer.
- Examining the shared library supplied by the third-party security software using the stand-alone test tool SSFTEST on the application server and frontend machines.
- Configuring and enabling the third-party security software on the R/3 application server (R/3 and security product settings).
- Configuring and enabling the third-party security software on the R/3 front ends.
- Testing the application server and frontend installations by running R/3 test reports.

2 The Test Procedure

This chapter describes details of the test procedure, to include the required steps and actions.

2.1 Preparing the Environment

Before you can perform the certification tests, you need to set up the test environment as described in the following sections.

2.1.1 Application Server

Install and configure an R/3 application server running R/3 Release 4.5B (or newer) on Microsoft Windows NT 4.0 (or newer) against a R/3 System and database Release 4.5B (or newer).

2.1.2 Front end

Installed and configure an R/3 Release 4.5B (or newer) front end running on Microsoft Windows NT 4.0 (or newer).

2.1.3 Installation Functional Test

1. Start the R/3 application server.
2. Check the console output and trace files (if required) to verify the successful establishment of the connection to the database and the correct status of the work and spool processes.
3. Start SAP Logon and configure an entry for the logon to the R/3 test system.
4. Start a SAPgui and log on to the R/3 test system.

2.2 Installing the Security Product on the Application Server

Complete any steps required to install the security product on the application server.

Note: As part of the security toolkit installation, you need to provide three SSF users as described in *Section 3.1.2*.

2.3 Installing the Security Product on the Frontend Machines

Complete any steps required to install the security product on the frontend machines.

Note: As part of the security toolkit installation, you need to provide three SSF users as described in *Section 3.1.2*.

Note: At this point, it is possible to test SSF with the stand-alone test tool as described in *Chapter 3.* The next two installation steps are necessary for the tests provided in Chapter 4.

2.4 Enabling SSF on the Frontend Machines

To enable SSF on the front ends, set the environment variable `SSF_LIBRARY_PATH` to contain the path to the security product library:

```
SSF_LIBRARY_PATH = <drive>:\path\to\your\ssflib.dll
```

Windows NT:	*Control Panel* ⇒ *System* ⇒ *Environment*
Windows 95:	edit `AUTOEXEC.BAT` and restart Windows 95
Windows 2003:	*Control Panel* ⇒ *System* ⇒ *Advanced* ⇒ *Environment Variables*

For more information on the frontend installation, see the *SSF User's Guide*.

2.5 Enabling SSF on the R/3 Application Server

1. Shut down the R/3 application server.
2. Add the following SSF-specific parameters to the R/3 application server's instance profile:

   ```
   ssf/ssfapi_lib= <drive>:\path\to\your\ssflib.dll
   ```

3. Start the R/3 application server.
4. Log on to the R/3 test system.

For more information on the installation, see the *SSF User's Guide*.

3 Using the Stand-Alone Test Tool SSFTEST

Most of the security toolkit's functions are tested using the stand-alone test tool SSFTEST.

You must execute the test on both the frontend machine and on the application server. The test is divided in two parts:

1. **Basic tests (required):** In this test, the basic operations are tested, including a test of all listed hash algorithms and symmetric encryption algorithms (see *Section 3.1.3*).
2. **PKCS#7 sample tests (recommended):** In this test, the security toolkit's compatibility with the PKCS#7 standard is tested using SSF samples provided by SAP. To test the verification of digital signatures, you can choose either RSA or DSA public keys (both with 1024 bit), each with the hash algorithms MD5 and SHA-1. You can (partially) disable these tests (see *Section 3.1.4*). After completing the tests, the results of the passed tests are included in the summary.

3.1 Installing SSFTEST

To install SSFTEST:

1. Copy all files from the directory `test` to an empty directory and change to that directory.

   Note that during the test, the test tool will create new files in this directory.

2. Before you can start SSFTEST (both on the application server and on the frontend machine), you have to configure the security product and change the parameters described in the following sections. Change them in the parameter file `ssftest.ini`. (Note that the settings on the frontend may be different than those on the application server.)

3.1.1 The Security Toolkit's SSF API Library

Modify the parameter `SSFLIBRARY`, so that it contains the location of your security toolkit's SSF API shared library.

```
SSFLIBRARY =        "c:\libssf.dll"
```

3.1.2 SSF Users

The test scenario uses three users for the various tests. Create these users with your security toolkit and adapt the lines in the parameter file that **refer to** the SSF profile (`UserProfile, UserPassword, UserId`) and private address book (`UserPab, UserPabPassword`) according to your security toolkit's requirements.

```
UserName =          "user1"
UserId =            "CN=user1, OU=SSF-TEST, O=SAP-AG, C=DE"
UserProfile =       "user1.pse"
UserPassword =      "u1"
UserPab =           "user1.pse"
UserPabPassword =   "u1"
...
```

Notes:

- You can use RSA or DSA as the asymmetric algorithm. The key length should be 1024 bits.
- If possible, the distinguished names of the users should be as follows. (The tests do not rely on those names, so you can change them (`UserId`) , if necessary.)
  - "`CN=user1, OU=SSF-TEST, O=SAP-AG, C=DE`"
  - "`CN=user2, OU=SSF-TEST, O=SAP-AG, C=DE`"
  - "`CN=user3, OU=SSF-TEST, O=SAP-AG, C=DE`"
- Depending on the security toolkit's requirements, you can use different users with different IDs (distinguished names) for the tests on the application server and on the front end.
- To test the function for verifying digital signatures, user2 should have access to user1's public key and user1 should have access to the all three users' public keys. (By default, the signatures are created without included certificates, that is, with `BIncCerts = 0`.)
- To test the function for creating a digital envelope, user1 must have access to all three users' public keys.
- For the final ABAP test (*Section 4.3*), a signature is created by the front end and verified by the application server. Therefore, for this test, there must be a PAB on the application server that is able to verify a frontend's signature.
- The tests in *Sections 4.2 and 4.3* require that the SSF profile and PAB can be accessed by the application server without entering a password. For a successful test, you have to administrate the application server correctly, for example, you need to install credentials for the application server.
- Do not modify the ssftest.par file without getting in contact with SAP !

3.1.3 Specify Algorithms

Specify the supported hash and symmetric encryption algorithms. These algorithms are tested and listed in the summary.

`HashAlgCount` and `SymEncrAlgCount` are the number of supported algorithms, followed by that number of lines of `HashAlgs` and `SymEncrAlgs`, respectively. Please notice that the hash algorithms MD5 and SHA1 are mandatory for this certification and must therefore be implemented.

For example:

```
HashAlgCount =     "4"
HashAlgs =         "MD5"
HashAlgs =         "SHA1"
HashAlgs =         "MD2"
HashAlgs =         "MD4"

SymEncrAlgCount =  "3"
SymEncrAlgs =      "DES-CBC"
SymEncrAlgs =      "TRIPLE-DES"
SymEncrAlgs =      "IDEA"
```

3.1.4 Sample Tests

The test tool is shipped with some sample SSF files for testing interoperability. You are not required to perform these tests, however, we recommend that you enable them. (The summery reports which of these tests you have executed.)

Disable all the tests by placing a semicolon „;“ at the beginning of the line `SAMPLE_TESTS`.

Alternatively, you can disable the following parts of the tests:

`SAMPLE_RSA`	Verify messages signed with RSA algorithm (and various hash algorithms)
`SAMPLE_DSA`	Verify messages signed with DSA algorithm (and various hash algorithms)
`USER4_INSTALLED`	The verification can be done both without the CA certificates installed and when these certificates are known.

For a complete test, add `sap_rsa_ca.cert` and `sap_dsa_ca.cert` (ASN.1 encoded X.509 certificates) or `*.p10` (PKCS #10 format: Certification Request) to any of your private address books and set PAB and password correctly for user4. (No „real“ user is required; only the PAB is used.)

```
        SAMPLE_TESTS =     "true"
; Do not comment the lines that contain the supported
; public key algorithms.
; Additionally, do not comment the line 'USER4_INSTALLED' if you
; have installed the respective CA certificates for user4
;
        SAMPLE_RSA =       "true"
        SAMPLE_DSA =       "true"
        USER4_INSTALLED =  "true"

; * user4: used to verify sap-samples; import certificate
        UserName =         "user4"
        UserPab =          "e:\PSE\rsa\user1.pse"
        UserPabPassword =  "u1"
```

3.2 Executing SSFTEST on the Frontend Machines

1. Complete the installation on the frontend machine as described in the previous section.
2. Start the SSFTEST tool on the frontend machine.

   ```
   ssftest > frontend-nt.log
   ```

3. Document in the test report any product-specific configuration or usage options that will affect the operation of SSFTEST.
4. **Save** or **archive** the output log **`frontend-nt.log`** created by the SSFTEST tool and interpret its contents according to *Section 3.4: Reading the Output Protocol of SSFTEST*.

For the BC-SSF certification, the product must pass all tests. In this case, a summary is printed. The final line of the summary is: `NO ERRORS FOUND`. For more information see *Chapter 3.4*.

3.3 Executing SSFTEST on the Application Server

1. Complete the installation on the application server as described in *Section 3.1*.

2. Start the SSFTEST tool on the application server

```
ssftest > appserv-nt.log
```

3. Document any product-specific configuration or usage options that will affect the operation of SSFTEST in the test report.
4. **Save** or **archive** the output protocol **`appserv-nt.log`** created by the SSFTEST tool and interpret its contents according to *Section 3.4: Reading the Output Protocol of SSFTEST*.

For the BC-SSF certification, the product must pass all tests. In this case, a summary is printed. The final line of the summary is: `NO ERRORS FOUND`. For more information see *Chapter 3.4*.

3.4 Reading the Output Protocol of SSFTEST

When the test of the security product completes successfully, SSFTEST compiles an output summary that contains the test results. It will look similar to the example shown below and contains the following information:

- Platform name
- Library name and version string
- Default hash algorithm
- Default symmetric encryption algorithm
- List of tested hash algorithms
- List of tested symmetric encryption algorithms
- List of passed sample tests

For the BC-SSF certification, the product must pass all tests. In this case, a summary is printed. The final line of the summary is: `NO ERRORS FOUND`.

```
**********************************************************************
**********************************************************************
********************** SSFTEST RESULT SUMMARY ************************
**********************************************************************
**********************************************************************

All SSF API tests successfully passed.

Hardware Platform:
        Microsoft Windows NT 4.0 (Build 1381)

loaded SSF library: libssf.XXX

    Version (length=47):
    Sample SSF Lib Version 1.0 for Windows 95/98/NT


Ssf Format: PKCS7
default hash algorithm: MD5
default symmetric encryption algorithm: DES-CBC

Tested hash algorithms:
        MD5
```

```
        SHA1
        MD2
        MD4

Tested symmetric encryption algorithms:
        DES-CBC
        TRIPLE-DES
        IDEA

Additional Sample Tests
        successfully passed the following tests:
        - SsfDigest
        - SsfVerify a RSA signature
        - SsfVerify a DSA signature
          (CA certificates installed: verification OK)

NO ERRORS FOUND.
```

3.5 Sample SSFTEST Output

The following is the complete sample output of SSFTEST:

```
ssftest version 1.0
  Michael Friedrich, (c) Copyright SAP AG, Walldorf
  Test program for the SSF API functions.

Timer resolution of QueryPerformanceCounter() is (at least)      0.008
millisec
1 second passed in    999.478 millisec.


======================================================================
    Current Date&Time :  Fri, 11-Jun-1999   11:31:01   GMT -02:00
    Operating System  :  Microsoft Windows NT
            -Release  :  4.0 (Build 1381)
    Hardware/Machine  :  x86  cpu_level=6, cpu_rev=0x0106
    Perf-Index (p-90) :  opt= 1.00   dbg= 2.00
    Timer Resolution: 0.008 millisec using "QueryPerformanceCounter()"
    Hostname          :  P21502
    Current user      :  d027442

======================================================================

        TITLE = SSF API Testsuite
        FILENAME = ssftest.par
        VERSION = 0.998          18 May 1999
        This testsuite is part of the SSF API Certification program.
```

```
  Loading the SSF API dynamic library...

===...SSF library libssf.dll loaded successfully.
===...SsfVersion loaded successfully.
===...SsfEncode loaded successfully.
===...SsfDecode loaded successfully.
===...SsfSign loaded successfully.
===...SsfVerify loaded successfully.
===...SsfEnvelope loaded successfully.
===...SsfDevelope loaded successfully.
===...SsfAddSign loaded successfully.
===...SsfDigest loaded successfully.
===...SsfDELSsfOctetstring loaded successfully.
===...SsfNEWSigRcpSsfInfo loaded successfully.
===...SsfDELSigRcpSsfInfo loaded successfully.
===...SsfINSSigRcpSsfInfo loaded successfully.
===...SsfDELSigRcpSsfInfoList loaded successfully.
===...SsfQueryProperties loaded successfully.
============================================================
Test Group 1: SsfVersion and SsfProperties
  SsfVersion
    Version (length=47):
    Sample SSF Lib Version 1.0 for Windows 95/98/NT

  SsfQueryProperties(PROPERTIES)
    QueryProperties: (length=25):
    FORMATS;HASHALGS;ENCRALGS
  SsfQueryProperties(...)
    PROPERTYDESCRIPTION: FORMATS
    QueryProperties: (length=5):
    PKCS7
    PROPERTYDESCRIPTION: HASHALGS
    QueryProperties: (length=26):
    MD2;MD4;MD5;SHA1;RIPEMD160
    PROPERTYDESCRIPTION: ENCRALGS
    QueryProperties: (length=23):
    DES-CBC;TRIPLE-DES;IDEA
  SsfQueryProperties(SSF_POPUPS)
  SsfQueryProperties(MISSING)
    Expected Error: SSF_API_NODATA (3)
============================================================
Test Group 2: SsfSign
  SsfSign_1_1 (first simple test)
  SsfSign_2_1 (small data)
  SsfSign_2_2 (medium-sized data)
  SsfSign_2_3 (large data)
  SsfSign_2_4 (extra large data)
  SsfSign_3_1 (don't include certificate)
  SsfSign_3_2 (certificates included)
  SsfSign_4_2 (two signers)
  SsfSign_4_1 (three signers)
  SsfSign_4_3 (two signers, certificates included)
  SsfSign_4_4 (two signers, certificates included, detached)
  SsfSign_5_1 (detached)
  SsfSign_5_2 (detached, with certificate)
  SsfSign_6_1 (wrong signer id)
    Expected Error: SSF_API_SIGNER_ERRORS (5)
  SsfSign_6_2 (wrong signer id: space)
    Expected Error: SSF_API_SIGNER_ERRORS (5)
  SsfSign_6_3 (wrong password)
    Expected Error: SSF_API_SIGNER_ERRORS (5)
============================================================
```

```
Test Group 3: SsfVerify
  SsfVerify_1_1 (first simple test)
  SsfVerify_1_2 (first simple test, verified by signer)
  SsfVerify_2_1 (small data)
  SsfVerify_2_2 (medium-sized data)
  SsfVerify_2_3 (large data)
  SsfVerify_2_4 (extra large data)
  SsfVerify_3_1 (successful, no included certificate, verified by
signer)
  SsfVerify_3_1 (successful, no included certificate)
  SsfVerify_3_2 (successful, certificates included)
  SsfVerify_3_3 (successful, certificates included)
  SsfVerify_3_4 (successful, no included certificate, BUseCert off)
  SsfVerify_3_5 (successful, certificate included, BUseCert off)
  SsfVerify_4_1 (two signers)
  SsfVerify_4_2 (several signers)
  SsfVerify_4_3 (two signers, certificates included)
  SsfVerify_4_4 (two signers, certificates included, detached)
  SsfVerify_5_1 (detached, originaldata given)
  SsfVerify_5_2 (detached, no originaldata)
    Expected Error: SSF_API_NODATA (3)
  SsfVerify_5_3 (detached, wrong originaldata given)
    Expected Error: SSF_API_SIGNER_ERRORS(5)
  SsfVerify_5_4 (not detached, correct originaldata given)
  SsfVerify_5_5 (not detached, wrong originaldata given)
    Expected Error: SSF_API_NODATA (3)
============================================================
Test Group 4: SsfEnvelope
  SsfEnvelope_1_1 (first simple test)
  SsfEnvelope_2_1 (small data)
  SsfEnvelope_2_2 (medium-sized data)
  SsfEnvelope_2_3 (large data)
  SsfEnvelope_2_3 (extra large data)
  SsfEnvelope_3_1 (self-enveloped)
  SsfEnvelope_4_1 (several recipients)
============================================================
Test Group 5: SsfDevelope
  SsfDevelope_1_1 (first simple test)
  SsfDevelope_1_2 (error if data is not encrypted for recipient)
    Expected Error: SSF_API_RECIPIENT_ERRORS (9)
  SsfDevelope_2_1 (small data)
  SsfDevelope_2_2 (medium-sized data)
  SsfDevelope_2_3 (large data)
  SsfDevelope_2_4 (extra large data)
  SsfDevelope_3_1 (self enveloped)
  SsfDevelope_4_1 (correct for first recipient in group of 2)
  SsfDevelope_4_2 (correct for second recipient in group of 2)
  SsfDevelope_4_3 (error for recipient not in group)
    Expected Error: SSF_API_RECIPIENT_ERRORS (9)
  SsfDevelope_4_4 (error if recipient unknown)
    Expected Error: SSF_API_RECIPIENT_ERRORS (9)
============================================================
Test Group 6: SsfAddSign
  SsfAddSign_1_1 (first simple test)
  SsfAddSign_1_2 (addsign after addsign)
  Verify SsfAddSign_1_1 (first simple test)
  Verify SsfAddSign_1_2 (addsign after addsign)
  SsfAddSign_2_1 (small data)
  SsfAddSign_2_2 (medium-sized data)
  SsfAddSign_2_3 (large data)
  SsfAddSign_2_4 (extra large data)
  SsfAddSign_3_1 (valid without certificates included)
  SsfAddSign_3_2 (certificates included)
```

```
  SsfAddSign_5_1 (detached)
  SsfAddSign_5_2 (detached)
    Expected Error: SSF_API_NODATA (3)
============================================================
Test Group 7: SsfDigest
  SsfDigest_1_1 (first simple test)
  SsfDigest_1_2 (once again)
  SsfDigest_2_1 (small data)
  SsfDigest_2_2 (medium-sized data)
  SsfDigest_2_3 (large data)
  SsfDigest_2_4 (extra large data)
  SsfDigest_3_1 (detached, small data)
  SsfDigest_3_2 (detached, medium-sized data)
============================================================
Test Group 8a: SsfComb (without 2nd encoding)
  SsfComb_1_1 (error if data is encoded but not signed)
    Expected Error: SSF_API_INVALID_FORMAT (2)
  SsfComb_1_2 (error if data is encoded but not enveloped)
    Expected Error: SSF_API_INVALID_FORMAT (2)
  SsfComb_2_1 (sign data)
  SsfComb_2_2 (sign signed data)
  SsfComb_2_3 (verify signed data with signed data content)
  SsfComb_2_4 (verify initial signed data)
  SsfComb_3_1 (sign initial data)
  SsfComb_3_2 (envelope signed data)
  SsfComb_3_3 (develope with signed data content)
  SsfComb_3_4 (verify initial signed data after develope)
  SsfComb_4_1 (envelope initial data)
  SsfComb_4_2 (envelope enveloped data)
  SsfComb_4_3 (develope with enveloped data content)
  SsfComb_4_4 (develope initial data)
  SsfComb_5_1 (envelope initial data)
  SsfComb_5_2 (sign enveloped data)
  SsfComb_5_3 (verify signed data with enveloped data content)
  SsfComb_5_4 (develope initial data)
============================================================
Test Group 8b: SsfComb (with encoding)
  SsfComb_2_1 (sign data)
  SsfComb_2_2 (sign signed data)
  SsfComb_2_3 (verify signed data with signed data content)
  SsfComb_2_4 (verify initial signed data)
  SsfComb_3_1 (sign initial data)
  SsfComb_3_2 (envelope signed data)
  SsfComb_3_3 (develope with signed data content)
  SsfComb_3_4 (verify initial signed data after develope)
  SsfComb_4_1 (envelope initial data)
  SsfComb_4_2 (envelope enveloped data)
  SsfComb_4_3 (develope with enveloped data content)
  SsfComb_4_4 (develope initial data)
  SsfComb_5_1 (envelope initial data)
  SsfComb_5_2 (sign enveloped data)
  SsfComb_5_3 (verify signed data with enveloped data content)
  SsfComb_5_4 (develope initial data)
============================================================
```

```
Test Group 9: Algorithm Tests
  SsfSign
      MD5  SHA1  MD2  MD4
  SsfVerify
      MD5  SHA1  MD2  MD4
  SsfAddSign
      MD5  SHA1  MD2  MD4
  SsfVerify
      MD5  SHA1  MD2  MD4
  SsfDigest
      MD5  SHA1  MD2  MD4
  SsfEnvelope
      DES-CBC  TRIPLE-DES  IDEA
  SsfDevelope
      DES-CBC  TRIPLE-DES  IDEA
============================================================
Test Group 10: Size Tests (1, 100, 1000 bytes)
       VERBOSE = 129
       LOOP = 3
  SsfDigest         70     172    1081
  (detached)        65      65      65

  SsfSign          596     695    1601
  (with certs)    1970    2068    2974
  (detached)       589     589     590
  (det., certs)   1965    1965    1965

  SsfEnvelope      253     353    1260
============================================================
Test Group 11: Timing Tests
       LOOP = 20
       TIMING = 1
  *** Small: 1024:     min             max             avg
  SsfSign           57.153          70.616          58.368 ms
  SsfVerify         95.164         103.019          96.400 ms
  SsfAddSign        56.894          62.767          57.492 ms
  SsfDigest          0.469           0.579           0.472 ms
  *** Medium-sized: 10240 bytes
  SsfSign           58.382          63.746          59.750 ms
  SsfVerify         96.090         104.253          98.362 ms
  SsfAddSign        58.111          63.590          59.353 ms
  SsfDigest          1.913           2.412           1.974 ms
  *** Large data: 102400 bytes
  SsfSign           85.024          93.268          86.087 ms
  SsfVerify        116.270         122.188         118.443 ms
  SsfAddSign        77.056          94.475          78.433 ms
  SsfDigest         27.217          32.112          27.622 ms

  *** Small: 1024:     min             max             avg
  SsfEnvelope       23.476          24.858          23.827 ms
  SsfDevelope       48.773          53.628          49.551 ms
  *** Medium-sized: 10240 bytes
  SsfEnvelope       28.137          42.740          28.672 ms
  SsfDevelope       53.333          62.148          53.945 ms
  *** Large data: 102400 bytes
  SsfEnvelope       87.292          94.107          88.176 ms
  SsfDevelope      109.484         120.301         111.318 ms
       LOOP = 0
============================================================
Test Group 12: Thread Tests
       LOOP = 5
       TIMING = 1
       THREADS = 50
```

```
  Thread_SsfSign_1 (small data)
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
          3088.601       19621.978        3189.738 ms
  Thread_SsfVerify_1 (small data)
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
          4944.764        5194.466        5004.481 ms
  Thread_SsfSign_2 (medium-sized data)
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
          3202.131        8942.093        3273.645 ms
  Thread_SsfVerify_2 (medium-sized data)
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
THREAD: All 50 threads finished, 0 with errors
          5032.938       12537.254        5192.299 ms
============================================================
Test Group 13a: Invalid Parameters (format)
        LOOP = 5
  SsfSign (invalid format)
    Expected Error: SSF_API_INVALID_FORMAT (2)
    FORMATDESCRIPTION: invalid format
    FORMATDESCRIPTION: format too short
    FORMATDESCRIPTION: format too long
    FORMATDESCRIPTION: empty format
    FORMATDESCRIPTION: format length -1
  SsfVerify (invalid format)
    Expected Error: SSF_API_INVALID_FORMAT (2)
    FORMATDESCRIPTION: invalid format
    FORMATDESCRIPTION: format too short
    FORMATDESCRIPTION: format too long
    FORMATDESCRIPTION: empty format
    FORMATDESCRIPTION: format length -1
  SsfAddSign (invalid format)
    Expected Error: SSF_API_INVALID_FORMAT (2)
    FORMATDESCRIPTION: invalid format
  SsfAddSign (invalid format)
    FORMATDESCRIPTION: format too short
  SsfAddSign (invalid format)
    FORMATDESCRIPTION: format too long
  SsfAddSign (invalid format)
    FORMATDESCRIPTION: empty format
  SsfAddSign (invalid format)
    FORMATDESCRIPTION: format length -1
  SsfDigest (error if invalid format)
    Expected Error: SSF_API_INVALID_FORMAT (2)
    FORMATDESCRIPTION: invalid format
    FORMATDESCRIPTION: format too short
    FORMATDESCRIPTION: format too long
    FORMATDESCRIPTION: empty format
    FORMATDESCRIPTION: format length -1
```

```
  SsfEnvelope (invalid format)
    Expected Error: SSF_API_INVALID_FORMAT (2)
    FORMATDESCRIPTION: invalid format
    FORMATDESCRIPTION: format too short
    FORMATDESCRIPTION: format too long
    FORMATDESCRIPTION: empty format
    FORMATDESCRIPTION: format length -1
  SsfDevelope (error if invalid format)
    Expected Error: SSF_API_INVALID_FORMAT (2)
    FORMATDESCRIPTION: invalid format
    FORMATDESCRIPTION: format too short
    FORMATDESCRIPTION: format too long
    FORMATDESCRIPTION: empty format
    FORMATDESCRIPTION: format length -1
============================================================
Test Group 13b: Invalid Parameters (no or wrong data)
        LOOP = 3
  SsfSign (no or wrong data)
    Expected Error: SSF_API_NO_DATA (3)
    DATADESCRIPTION: error if no data
    DATADESCRIPTION: error if zero data length
    DATADESCRIPTION: error if data length -1 specified
  SsfVerify (no or wrong data)
    Expected Error: SSF_API_NO_DATA (3)
    DATADESCRIPTION: error if no data
    DATADESCRIPTION: error if zero data length
    DATADESCRIPTION: error if data length -1 specified
  SsfEnvelope (no or wrong data)
    Expected Error: SSF_API_NO_DATA (3)
    DATADESCRIPTION: error if no data
    DATADESCRIPTION: error if zero data length
    DATADESCRIPTION: error if data length -1 specified
  SsfDevelope (no or wrong data)
    Expected Error: SSF_API_NO_DATA (3)
    DATADESCRIPTION: error if no data
    DATADESCRIPTION: error if zero data length
    DATADESCRIPTION: error if data length -1 specified
============================================================
Test Group 13c: Invalid Parameters (no or wrong data, no SsfEncode)
        LOOP = 3
  SsfSign (no or wrong data)
    Expected Error: SSF_API_NO_DATA (3)
    DATADESCRIPTION: error if no data
    DATADESCRIPTION: error if zero data length
    DATADESCRIPTION: error if data length -1 specified
  SsfEnvelope (no or wrong data)
    Expected Error: SSF_API_NO_DATA (3)
    DATADESCRIPTION: error if no data
    DATADESCRIPTION: error if zero data length
    DATADESCRIPTION: error if data length -1 specified
============================================================
Test Group 13d: Invalid Parameters (no encoding)
  SsfSign (error if input data is not encoded)
    Expected Error: SSF_API_DECODE_FAILED (12)
  SsfVerify (error if input data is not encoded)
    Expected Error: SSF_API_DECODE_FAILED (12)
  SsfEnvelope (error if input data is not encoded)
    Expected Error: SSF_API_DECODE_FAILED (12)
  SsfDevelope (error if data is not encoded data)
    Expected Error: SSF_API_DECODE_FAILED (12)
============================================================
```

```
Test Group 13e: Invalid Parameters (wrong ids)
  SsfSign (error if recipient id length too short)
    Expected Error: SSF_API_SIGNER_ERRORS (5)
  SsfSign (error if recipient id length too long)
    Expected Error: SSF_API_SIGNER_ERRORS (5)
  SsfEnvelope (error if recipient id unknown or id length too short)
    Expected Error: SSF_API_RECIPIENT_ERRORS (9)
  SsfEnvelope (error if recipient id length too long)
    Expected Error: SSF_API_RECIPIENT_ERRORS (9)
============================================================
Test Group 13f: Invalid Parameters (strange profile names)
  SsfSign (strange profile names)
    Expected Error: return code ignored
    PROFILEDESCRIPTION: (null)
    PROFILEDESCRIPTION:
    PROFILEDESCRIPTION: /
    PROFILEDESCRIPTION: \
    PROFILEDESCRIPTION: \\
    PROFILEDESCRIPTION: :
    PROFILEDESCRIPTION: .
    PROFILEDESCRIPTION: @
    PROFILEDESCRIPTION: *
    PROFILEDESCRIPTION: ?
    PROFILEDESCRIPTION: ssftest.par
    PROFILEDESCRIPTION: null
============================================================
Test Group 14: Samples for SsfDigest
  SsfDigest (compare digest)
    ALGDESCRIPTION: MD5
    ALGDESCRIPTION: SHA1
============================================================
Test Group 15a: Samples for SsfVerify (RSA)
RSA (CA unknown)
    SsfVerify_1 (with certificate)
      ALGDESCRIPTION: MD5
      ALGDESCRIPTION: SHA1
    SsfVerify_2 (detached, with certificate)
      ALGDESCRIPTION: MD5
      ALGDESCRIPTION: SHA1
RSA (CA is known)
    SsfVerify_1 (with certificate)
      ALGDESCRIPTION: MD5
      ALGDESCRIPTION: SHA1
    SsfVerify_2 (detached, with certificate)
      ALGDESCRIPTION: MD5
      ALGDESCRIPTION: SHA1
============================================================
Test Group 15b: Samples for SsfVerify (DSA)
DSA (CA unknown)
    SsfVerify_1 (with certificate)
      ALGDESCRIPTION: MD5
      ALGDESCRIPTION: SHA1
    SsfVerify_2 (detached, with certificate)
      ALGDESCRIPTION: MD5
      ALGDESCRIPTION: SHA1
DSA (CA is known)
    SsfVerify_1 (with certificate)
      ALGDESCRIPTION: MD5
      ALGDESCRIPTION: SHA1
    SsfVerify_2 (detached, with certificate)
      ALGDESCRIPTION: MD5
      ALGDESCRIPTION: SHA1
```

```
**********************************************************************
**********************************************************************
********************** SSFTEST RESULT SUMMARY ************************
**********************************************************************
**********************************************************************

All SSF API tests successfully passed.

Hardware Platform:
        Microsoft Windows NT 4.0 (Build 1381)

loaded SSF library: libssf.XXX

    Version (length=47):
    Sample SSF Lib Version 1.0 for Windows 95/98/NT


Ssf Format: PKCS7
default hash algorithm: MD5
default symmetric encryption algorithm: DES-CBC

Tested hash algorithms:
        MD5
        SHA1
        MD2
        MD4

Tested symmetric encryption algorithms:
        DES-CBC
        TRIPLE-DES
        IDEA

Additional Sample Tests
        successfully passed the following tests:
        - SsfDigest
        - SsfVerify a RSA signature
        - SsfVerify a DSA signature
          (CA certificates installed: verification OK)

NO ERRORS FOUND.
```

4 R/3 Test Report SSFTEST

The stand-alone tool SSFTEST tests most of the security toolkit's functions. The following R/3 tests focus on the correct installation for R/3.

This tests are divided in three parts:

1. Tests on the front end
2. Tests on the application server
3. Combined tests (signing on the front end, verifying on the application server)

4.1 Testing SSF on the Frontend Machines

To test SSF on the front end:

1. Start a SAPgui and log on to the R/3 test system (in English).
2. Call Transaction SE38 and start report **SSFTEST**.
3. Enter the appropriate values in the following fields. (Refer to *Section 3.1.2* for more information about the users.)

   Note: As of Release 4.6, this report is available as report `SSF01` in the standard delivery. A slightly modified version has been transported to the 4.5B test systems of ICC.

   - **RFC Destination:** Enter the value **SAP_SSFATGUI**.
   - **Signer**: Specify the *ID*, *Profile* and *Password* of the signer. (Use **user1** as described in *Section 3.1.2*.)
   - **Recipient**: Specify the *PAB* and *PAB password* of the recipient. (Use **user2** as described in *Section 3.1.2*.)
   - **Hash algorithm:** Change the algorithm for *SsfDigest* (default: `MD5`)

4. Enable *Execute immediately* and *Combine multiple tests*.
5. Start the report with *Execute* (`F8`).

   The system displays a listing with the results of the various tests. The last line should state that no errors have occurred.

6. **Save** or **archive** the output list **frontend2-nt.log** created by the SSFTEST report and interpret its contents according to *Section 4.4*.

4.2 Testing SSF on the Application Server

This test is similar to the previous one, however, in this test, the cryptographic operations are performed on the application server. After performing the test on the front end (see the previous section):

1. Go *Back* (F3)
2. Change the following fields:
   - **RFC Destination:** Clear this field to indicate that the R/3 kernel should perform the operations.
   - **Security Toolkit:** If you have installed multiple security toolkits, specify the security toolkit's name in this field. Leave it empty to use the default security toolkit (loaded with `ssf/ssfapi_lib`). For more information, see the *SSF User's Guide*.
   - **Signer:** If necessary, change the user's *ID* and *Profile* (application server: **`user1`**). Clear the *Password* field.
   - **Recipient:** If necessary, change the *PAB* (application server: **`user2`**). Clear the *PAB Password* field.

   Note: On the application server, you must **leave the password fields blank**. Instead of using passwords, the application server accesses the profile and PAB using other means (for example, using credentials).
3. Start the report with *Execute* (F8).

   The system displays a listing with the results of the various tests. The last line should state that no errors have occurred.
4. **Save** or **archive** the output list **`appserv2-nt.log`** created by the SSFTEST report and interpret its contents according to *Section 4.3*.

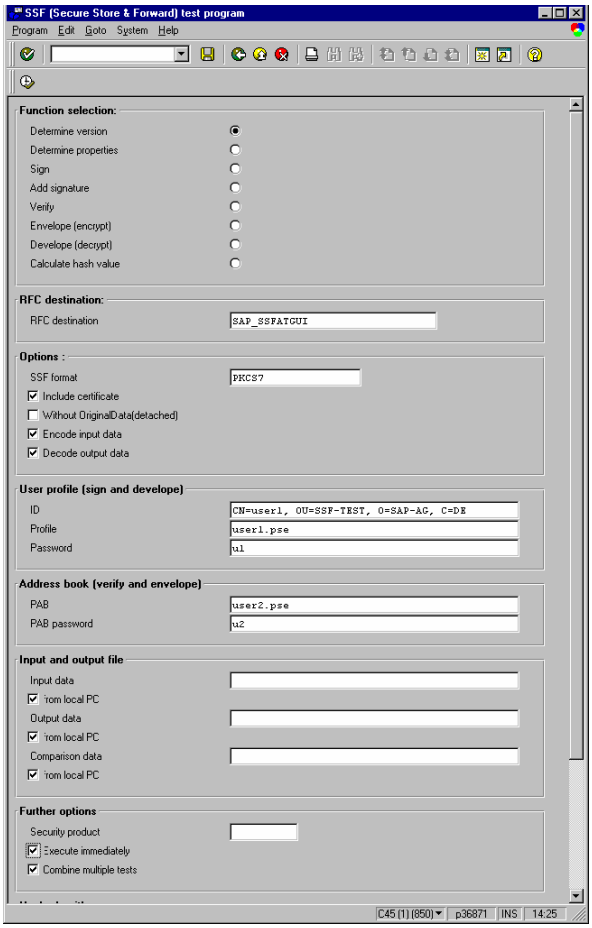


Figure 1: SSFTEST on the front end

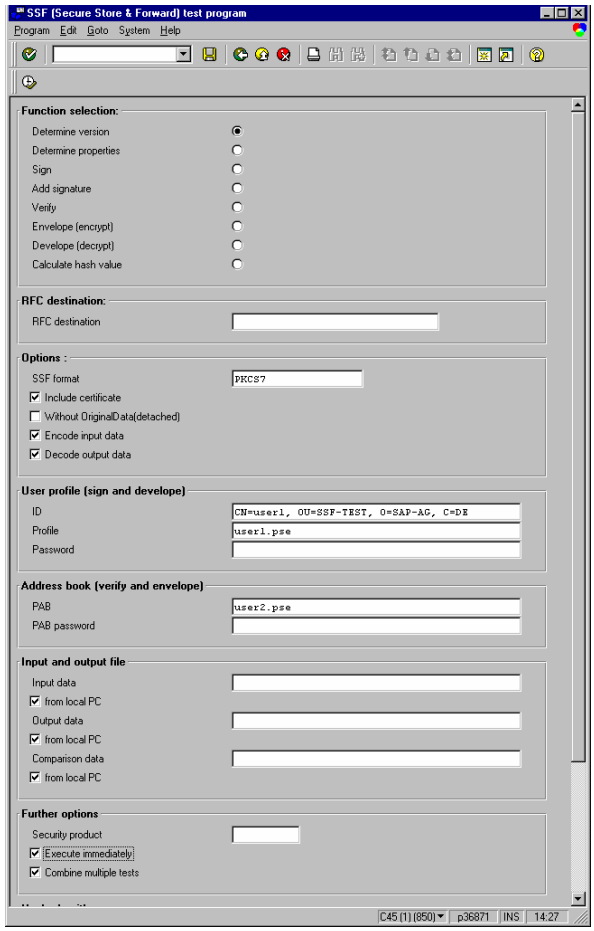


Figure 2: SSFTEST on the application server

4.3 Combined Front End and Application Server Test

This final test checks the correct installation and interoperability of the frontend SSF and the application server SSF. A document is signed on the front end and then verified by the application server. Perform the following:

1. Restart report SSFTEST (for example, use Transaction SE38).
2. Enter the appropriate values in the following fields. (Refer to *Section 3.1.2* for information about the users).
   - **Function selection:** Select the operation *Sign*.
   - **RFC Destination:** Enter the value **`SAP_SSFATGUI`** (default value).
   - **Signer**: Specify the *ID*, *Profile* and *Password* of the signer. (Use frontend **`user1`** as described in section 3.1.2.)
3. Enable *Execute immediately* (default value).
4. Disable *Combine multiple tests* (default value).
5. Start the report with *Execute* (`F8`).

   The system displays a list containing the results.
6. Refer to the resulting list to make sure that the signature succeeded.

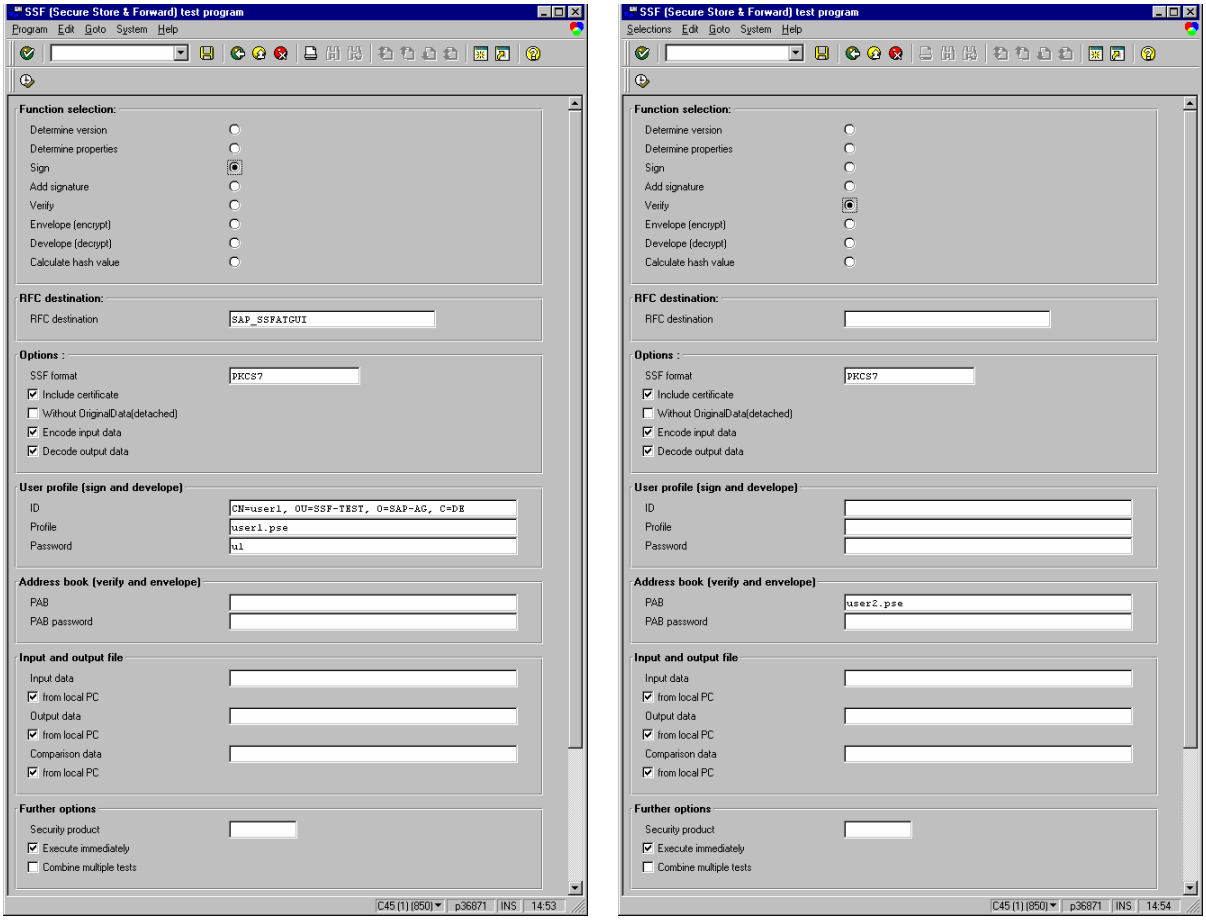


Figure 3: Combined test, part 1 and 2

7. Proceed with the verification by choosing *Next Operation* (F8) and change the following fields:

   - **RFC Destination:** Clear this field to indicate that the R/3 kernel should perform the operations.
   - **Security Toolkit:** If you have installed multiple security toolkits, specify the security toolkit's name in this field. Leave it empty to use the default security toolkit (loaded with `ssf/ssfapi_lib`).
   - **Recipient**: Specify the *PAB* of the recipient. (Use application server **`user2`** as described in *Section 3.1.2*.) Note that this is the application server's PAB. As already mentioned in the previous section, leave the password field blank.

8. Start the verification with *Execute* (F8).

   The system displays a list containing the results.

9. Refer to the resulting list to make sure that the signature succeeded.

4.4 Reading the Output Protocol of Report SSFTEST

When executed, the test report SSFTEST generates an output list similar to the one shown below.

Note: The sample list is from a frontend test. For the application server test, the title `RFC destination` is replaced by `(at application server)`.

For the BC-SSF certification, all tests must be passed with `SSF_API_OK` and the final line must be: `All tests successfully passed`.

```
17.05.1999                          SSF Test program                1
------------------------------------------------------------------------
Version              RFC destination      SAP_SSFATGUI
------------------------------------------------------------------------

Result:  SSF_API_OK


Version information:                                          47

Sample SSF Lib Version 1.0 for Windows 95/98/NT

17.05.1999                          SSF Test program                2
------------------------------------------------------------------------
Property             RFC destination      SAP_SSFATGUI
------------------------------------------------------------------------

Result:  SSF_API_OK

PROPERTIES
       FORMATS
       HASHALGS
       ENCRALGS
```

```
------------------------------------------------------------------------
FORMATS
       PKCS7
------------------------------------------------------------------------
HASHALGS
       MD2
       MD4
       MD5
       SHA1
       RIPEMD160
------------------------------------------------------------------------
ENCRALGS
       DES-CBC
       TRIPLE-DES
       IDEA

17.05.1999                          SSF Test program                  3
------------------------------------------------------------------------
Sign                 RFC destination      SAP_SSFATGUI
------------------------------------------------------------------------
Input data:                                                      21

         This is a sample text

User profile (sign and develope)

         CN=user1, OU=SSF-TEST, O=SAP-AG, C=DE
         c:\test\pse\user1.pse

Result:  SSF_API_OK

Results for the signer:

         CN=user1, OU=SSF-TEST, O=SAP-AG, C=DE
         SSF_API_SIGNER_OR_RECIPIENT_OK

Output data:                                                  1.016

         0.....*.H.........0......1.0...*.H.......0$..*.H.........Th
is is a sample         SSF-TEST1.0...U....user10..0...*.H...........0
.......g.8p... ..Vg=......R..          ..U..........0...U.......0.0...
*.H...........p.x....#..#w.Ar].,.8H...d..E         0...*.H........Y0.
..*.H......1...*.H......0...*.H......1...990517130858Z0..

17.05.1999                          SSF Test program                  4
------------------------------------------------------------------------
Verify               RFC destination      SAP_SSFATGUI
------------------------------------------------------------------------
Input data:                                                   1.016

         0.....*.H.........0......1.0...*.H.......0$..*.H.........Th
is is a sample         SF-TEST1.0...U....user10..0...*.H...........0.
......g.8p... ..Vg=......R..          ..U..........0...U.......0.0...*
.H...........p.x....#..#w.Ar].,.8H...d..E         0...*.H........Y0..
.*.H......1...*.H......0...*.H......1...990517130858Z0..

Address book for signer:

         c:\test\pse\user2.pse

Result:  SSF_API_OK
```

```
Results of the digital signature check:

         CN=user1, OU=SSF-TEST, O=SAP-AG, C=DE
         SigningTime= Mon May 17 15:08:58 1999 (UTCTime:
990517130858Z)
         SSF_API_SIGNER_OR_RECIPIENT_OK

Output data:                                                  21

         This is a sample text

17.05.1999                          SSF Test program                 5
------------------------------------------------------------------------
Hash Alg.            RFC destination      SAP_SSFATGUI
------------------------------------------------------------------------
Input data:                                                   21

         This is a sample text

Result:  SSF_API_OK

Output data:                                                  90

         0X..*.H.......K0I...0...*.H.......0$..*.H.........This is a
sample text...

17.05.1999                          SSF Test program                 6
------------------------------------------------------------------------
Envelope             RFC destination      SAP_SSFATGUI
------------------------------------------------------------------------
Input data:                                                   21

         This is a sample text

Recipient:

         CN=user1, OU=SSF-TEST, O=SAP-AG, C=DE

Address book (verify and envelope)

         c:\test\pse\user2.pse

Result:  SSF_API_OK


Results for recipients:

         CN=user1, OU=SSF-TEST, O=SAP-AG, C=DE
         SSF_API_SIGNER_OR_RECIPIENT_OK


Output data:                                                 308

         0..0..*.H.........!0......1..0.....0C0.1.0...U....DE1.0...U..
..SAP-AG1.0...         .H......0...+......ZYD.........H......$...T.....
6.U...
```

```
17.05.1999                          SSF Test program                  7
------------------------------------------------------------------------
Develope             RFC destination      SAP_SSFATGUI
------------------------------------------------------------------------
Input data:                                                      308

         0..0..*.H.........!0......1..0.....0C0.1.0...U....DE1.0...U..
..SAP-AG1.0...          .H......0...+......ZYD.........H......$...T.....
6.U...

Develope for:

         CN=user1, OU=SSF-TEST, O=SAP-AG, C=DE
         c:\test\pse\user1.pse

Result:  SSF_API_OK


Results for recipients:

         CN=user1, OU=SSF-TEST, O=SAP-AG, C=DE
         SSF_API_SIGNER_OR_RECIPIENT_OK

Output data:                                                      21

         This is a sample text

------------------------------------------------------------------------
All tests successfully passed.
```