Segurança ABAP – Parte IV – Secure Store & Forward (SSF)

28/03/2013 09:00

O quê? Post novo da série de Segurança? O MUNDO ACABOU???

NÃO, CARO AMIGO ZUMBIZÓIDE! Cá estamos nós de volta (eu no caso), dessa vez pra falar da parte que talvez é a mais complexa, porém a que eu achei mais divertida, do mundinho de segurança ABAP: os mecanismos de assinatura digital e criptografia que podem ser usados através da interface de segurança do SAP, ***Secure Store & Forward*** **(BC-SEC-SSF)**, que chamarei apenas de SSF.

> AQUI MORTO VIVO NÃO PASSA NÃO TÁ
> PENSANDO QUE É O QUE

Antes de mais nada™

Em primeiro lugar, é importante ficar claro aqui que **não existe API de criptografia implementada puramente em ABAP** . Procurando no SCN, aparecem centenas de resultados que indicam centenas de funções e classes como "soluções" para criptografar dados em ABAP. Eis algumas delas:

- FIEB_PASSWORD_ENCRYPT
- ENCODE_SLDPWD_BASE64
- HTTP_SCRAMBLE
- OFX_ALS_PASSWORD_ENCRYPT
- K_ENCRYPT_RESOURCE
- SSFS_ENCODE_TEXT
- SSFC_BASE64_ENCODE
- MAKE_PASSWORD_UNREADABLE
- CL_BSP_UTILITY=>ENCODE_STRING
- CL_HARD_WIRED_ENCRYPTOR
- métodos ENCODE* da interface IF_HTTP_UTILITY

Fuçando no Google eu achei inclusive um projeto no Google Code onde o cara implementou o algoritmo RSA em ABAP para usar com *strings* de códigos ASCII. Essas "soluções", por mais que pareçam boas, tem pelo menos um dos seguintes problemas:

- o algoritmo de criptografia não atende padrões regulatórios (PKCS, IETF, ASN.1, leis internacionais de proteção da informação, etc.);
- o algoritmo não é de criptografia de chave pública;
- não está sendo usado nenhum algoritmo de encriptação, mas sim de codificação (Base64, UTF-8, etc.).

Entendendo o SSF

Antes de sair pulando pro código, é bom entender como funciona a ~~física quântica~~ infraestrutura por trás da SSF.

O propósito do SSF é **proteger documentos dentro do SAP** através de **assinaturas digitais** e **envelopes digitais**. Não vou explicar nesse post como funcionam os mecanismos da criptografia, mesmo porque são coisas que podem ser aprendidas facilmente na Wikipedia (principalmente funções de hash (inglês), algoritmos de chave simétrica (inglês), criptografia de chave pública (inglês), infraestrutura de chaves públicas (inglês)), porém cabe definir aqui de maneira geral como esses processos funcionam **para o SAP**.

Assinatura digital é o processo de autenticação de documentos digitais que ocorre da seguinte maneira:

1. O autor de um documento calcula o *hash code* do mesmo e encripta esse código resultante usando sua chave privada;
2. Então, o autor anexa o código criptografado ao documento original ("assina" o documento) e essa mensagem é enviada ao destinatário;
3. O destinatário recebe a mensagem e também calcula o *hash code* do documento original (com o mesmo algoritmo de *hash* usado pelo autor);
4. O destinatário desencripta o trecho "assinado" da mensagem usando a chave pública do autor, obtendo assim o *hash code* que foi calculado inicialmente pelo autor;

5. Finalmente, o destinatário compara os dois códigos obtidos. Se forem diferentes, isso significa que a) a mensagem foi alterada após ser assinada ou b) a assinatura não foi gerada com a chave privada correspondente à chave pública usada na desencriptação.

Já os **envelopes digitais** funcionam de maneira semelhante, porém "ao contrário". A ideia aqui é proteger um documento usando criptografia híbrida para que apenas o destinatário possa visualizá-lo, da seguinte maneira:

1. O autor encripta o documento original usando um algoritmo de criptografia simétrica;
2. O autor anexa a chave da criptografia simétrica ao documento criptografado e encripta todo esse conteúdo novamente, mas dessa vez usando a chave pública do destinatário, e envia a mensagem;
3. O destinatário recebe a mensagem, desencripta-a usando sua chave privada, e desencripta o documento usando a chave simétrica inclusa.

A função da SSF é a mesma da *Virus Scan Interface* (que eu abordei no post anterior dessa série): o ECC em si não provê os serviços, mas fornece a interface para um produto externo, que deve ser escolhido entre os fornecidos por parceiros certificados pela SAP.

Toda instalação do ECC (inclusive as versões *trial*) já vem com um provedor de serviços padrão **apenas para a assinatura digital do servidor de aplicação**, que é chamado de SAP *Security Library* (SAPSECULIB). Se você pretende usar serviços de criptografia ou de assinatura digital mais abrangentes, você deve usar um produto externo. A SAP disponibiliza para download gratuito no Marketplace uma biblioteca de serviços de criptografia chamada SAP *Cryptographic Library* (SAPCRYPTOLIB) – porém, esta biblioteca está sujeita às leis de exportação de tecnologia criptográfica da Alemanha e sendo assim não está disponível para todos os clientes. A SAPCRYPTOLIB contém, além de algoritmos de criptografia, todas as funções já disponíveis na SAPSECULIB.

Dependendo do produto externo que for usado, teremos uma infraestrutura de chaves públicas diferente. Quando lidamos com SAPSECULIB / SAPCRYPTOLIB, as informações de chave pública e privada do servidor ficam guardadas em arquivos chamados de *Personal Security Environments* (PSEs). Em instalações de outros produtos que utilizem *smart cards*, por exemplo, a informação de chave privada do usuário ficaria dentro do *smart card*, enquanto a informação de chave pública ficaria disponível no servidor de aplicação.

OK, mas cadê o código?

Pra saber como usar as funções do SSF, o melhor é consultar o ***SSF Programmer's Guide*** (obviamente em inglês). As funções estão no grupo SSFG e as principais são as seguintes:

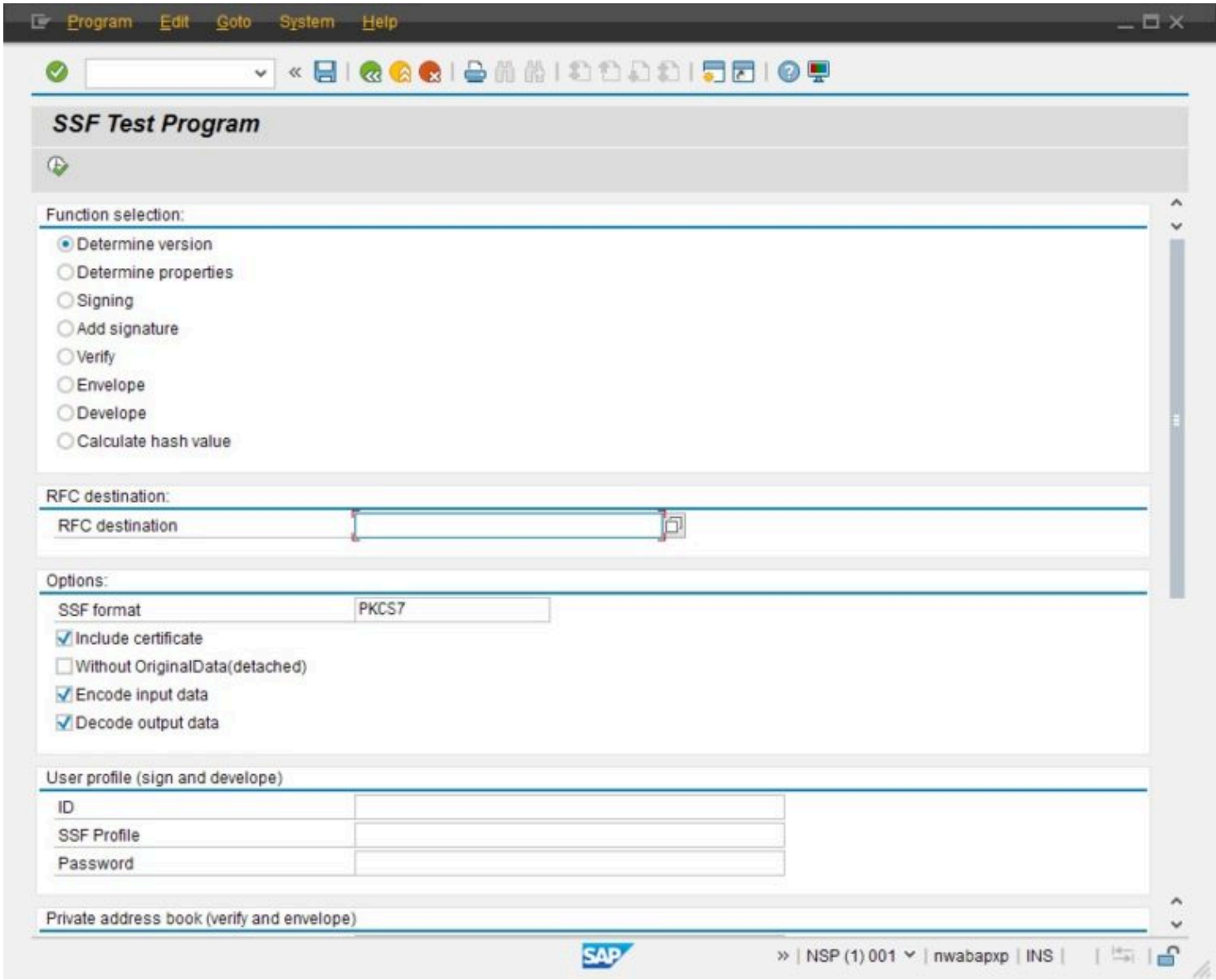
- SSF_SIGN / SSF_KRN_SIGN: servem para assinar um documento com uma chave privada;
- SSF_VERIFY / SSF_KRN_VERIFY: servem para verificar uma assinatura em um documento;
- SSF_ENVELOPE / SSF_KRN_ENVELOPE: criam um envelope digital com um documento;
- SSF_DEVELOPE / SSF_KRN_DEVELOPE: decodificam um envelope digital.

As funções SSF_* utilizam destinos RFC, enquanto as funções SSF_KRN_* executam comandos diretamente no *kernel* e portanto são mais rápidas.

Quando é necessária a intervenção do usuário, algumas funções do SSF exibem telas específicas para entrada de usuário e senha, exibição do texto que será assinado, seleção de arquivo com a chave privada, etc.

Assim como acontece com a *Virus Scan Interface*, o Basis tem o trabalho maior de configurar todo o ambiente da SSF e passar os parâmetros para que as funções ABAP possam ser usadas.

Deu um trabalho NERVOSO, mas eu consegui configurar o SSF na minha instalação *trial* do Netweaver e testar as funções. Infelizmente eu não consegui configurar assinaturas / envelopes para usuários específicos, mas pelo menos foi possível usar assinaturas e envelopes do servidor. Depois de instalado, é possível testar as funções e configurações nos *reports* SSF01 e SSF02.



Esse é o SSF02, mesma coisa que o SSF01 só que com menos parâmetros preenchidos

Para fazer os testes, eu criei um arquivo .TXT com o seguinte conteúdo:

```
ABAP Zombie
Prevenindo Consultores de Virarem Zumbis
```

Vamos ver primeiro o resultado do ***Signing***:

```
Sign (on application server)

Input data: 53

ABAP Zombie..Prevenindo Consultores de Virarem Zumbis

User profile (sign and develope)

CN=nwabapxp, OU=IDEMOSYSTEM, OU=SAP Web AS, O=SAP Trust Community, C=DE
SAPSSLS.pse

Time: 19 ms

Result: SSF_API_OK

Results for the signatory:

CN=nwabapxp, OU=IDEMOSYSTEM, OU=SAP Web AS, O=SAP Trust Community, C=DE
SSF_API_SIGNER_OR_RECIPIENT_OK

Output data: 492
```

```
0#.#..*#H##....##.#0#.#...1.0...*#H##.....0D..*#H##....#7.5ABAP Zombie..Prevenindo Consultores de Virarem Zumbis1#.x0#.t.
```

Aqui ainda é possível ver o conteúdo original do documento, porém acrescido da informação gerada pela assinatura com a chave privada do servidor.

Logo após o ***Signing***, é possível executar o ***Verify*** com os dados obtidos:

```
Verify (on application server)

Input data: 492

0#.#..*#H##....##.#0#.#...1.0...*#H##.....0D..*#H##....#7.5ABAP Zombie..Prevenindo Consultores de Virarem Zumbis1#.x0#.t.

Address Book for Signatory:

SAPSSLS.pse

Time: 5 ms

Result: SSF_API_OK

Results of the digital signature check:

CN=nwabapxp, OU=IDEMOSYSTEM, OU=SAP Web AS, O=SAP Trust Community, C=DE
SigningTime= Wed Mar 27 09:37:28 2013 (UTCTime: 130327123728Z)
SSF_API_SIGNER_OR_RECIPIENT_OK

Output data: 53

ABAP Zombie..Prevenindo Consultores de Virarem Zumbis
```

Aqui vemos a informação de volta ao formato original e sabemos que a assinatura do servidor e o documento estão corretos devido ao retorno SSF_API_SIGNER_OR_RECIPIENT_OK da função.

Vamos tentar agora o ***Envelope***:

```
Envelope (on application server)

Input data: 53

ABAP Zombie..Prevenindo Consultores de Virarem Zumbis

Recipient:

CN=nwabapxp, OU=IDEMOSYSTEM, OU=SAP Web AS, O=SAP Trust Community, C=DE

Private address book (verify and envelope)

SAPSSLS.pse

Time: 8 ms

Result: SSF_API_OK

Results for recipients:

CN=nwabapxp, OU=IDEMOSYSTEM, OU=SAP Web AS, O=SAP Trust Community, C=DE
SSF_API_SIGNER_OR_RECIPIENT_OK

Output data: 391

0#.#..*#H##....##.t0#.p...1#..0#.....0t0i1.0...U....DE1.0...U....SAP Trust Community1.0...U....SAP Web AS1.0...U....IDEMOS
```

Aqui não é possível identificar a informação original, que está encriptada com o algoritmo simétrico e com a chave pública do destinatário (que nesse caso é o próprio autor, o servidor).

Logo após a execução do ***Envelope***, o programa já mostra o resultado do ***Develope***:

```
Develope (on application server)

Input data: 391

0#.#..*#H##....##.t0#.p...1#..0#.....0t0i1.0...U....DE1.0...U....SAP Trust Community1.0...U....SAP Web AS1.0...U....IDEMOS

Develope for:

CN=nwabapxp, OU=IDEMOSYSTEM, OU=SAP Web AS, O=SAP Trust Community, C=DE
SAPSSLS.pse
```

```
Time: 9 ms

Result: SSF_API_OK

Results for recipients:

CN=nwabapxp, OU=IDEMOSYSTEM, OU=SAP Web AS, O=SAP Trust Community, C=DE
SSF_API_SIGNER_OR_RECIPIENT_OK

Output data: 53

ABAP Zombie..Prevenindo Consultores de Virarem Zumbis
```

E aqui temos a informação original desencriptada de volta.

Os padrões usados nestes testes são:

- PKCS7 para a sintaxe da mensagem gerada,
- RSA para a criptografia de chave pública,
- DES-CBC para a criptografia simétrica, e
- SHA1 para o hash.

É isso que eu tinha pra falar sobre SSF. Estou deixando muitos detalhes de fora, como instalação, configuração, certificados, mas tudo isso tem mais a ver com Basis do que com ABAP, e dá pra conferir isso tudo no próprio *help* da SAP sobre o assunto.

Tá confuso? Escrevi groselha? Faltou coisa? Manda um comentário ou um email. Até a próxima.

Comentários

Diogenes Kaue — *11/06/2018 08:53*

Bom dia Leo Schmidt, tudo bem ?

Estou atualmente em uma consultoria SAP, tenho dois anos de desenvolvimento ABAP, recentemente a consultoria me pediu para procurar formas de proteger o código que venho desenvolvendo nos clientes.
Li em alguns post pela internet em que através do trecho abaixo consigo deixar o código invisível, para o editor contudo não consigo dar manutenção após a utilização do mesmo.

****
*@#@@[SAP]
****

Tem alguma sugestão para proteger o código, criptografar ?

> **Leo Schmidt** — *11/06/2018 10:31*
>
> Bom dia Diogenes, td certo cara! Valeu pelo contato!
>
> O teu caso é bastante complicado. O que eu expus no post serve pra um caso muito específico, que seria a criptografia de documentos dentro do próprio SAP. Pra código fonte, eu acredito que não tem uma solução de criptografia muito prática. O que eu consigo pensar no teu cenário é usar essa alternativa que vc mesmo mencionou, e armazenar o código visível numa cópia do ambiente de DEV onde vc possa desenvolver livremente. Infelizmente teria um passo a mais, que seria transportar o código desse sistema pro sistema do cliente e deixá-lo invisível, mas acho que é o preço a se pagar pra manter os direitos do código.
>
> Abraços!

Luiz Guilherme Cerqueira Campos — *23/09/2016 13:21*

Boa tarde senhores,

Primeiramente parabéns pelo post, show de bola, referente a algo tão incomum e que pouquíssimos ABAPers conheçam – porém, de extrema importância. Estou com um requerimento onde faço uma chama http para utilizar um serviço, consegui realizar. O problema é que os dados que encorpam o JSON, são dados sensíveis, precisamos criptografar este json com a chave pública disponibilizada pelo serviço que consegue decifra-la pela chave privada. Importamos essa chave no PSE? Tentei importar o certificado com a chave, porém sem sucesso. Teria algum exemplo para nos ajudar?

Atenciosamente,

> **Leo Schmidt** — *23/09/2016 16:02*
>
> Olá Luiz, obrigado pelo elogio.
>
> Cara, pelo pouco que eu sei o procedimento seria isso mesmo: importar o certificado com a chave pública pela STRUST e então usar a SSF_ENVELOPE pra criptografar.
>
> Os programas que eu mencionei no post (SSF01 / SSF02) são os melhores exemplos que eu posso te dar, porque eles demonstram tudo o que as funções de criptografia podem fazer.
>
> Sei que não é muito, mas espero que isso ajude.
>
> []s,
> Leo

Espindola — *21/10/2014 08:44*

Bom Dia
Com relação ao seu guia de Criptografia. Sensacional muito bem detalhado.
Consegui gerar a assinatura, porém a criptografia não funcionou. Vc lembra quais parametros colocou ??
Obrigado
Luiz

> **Leo Schmidt** — *23/09/2016 16:13*
>
> Luiz, sinto muito por responder só agora, mas não sei te dizer.
>
> Eu lembro vagamente de começar pelo padrão da SSF02 e ir ajustando a execução com os parâmetros até fazer funcionar.
>
> Mais uma vez, desculpe pela negligência 😕
>
> Leo