



PL/PDF OffX v5

PL/PDF Encrypt

v5.31

Setup, supported algorithms, backward compatibility, and AES-256 example

1. Purpose and scope

PL/PDF v5.31 extends the built-in PDF encryption support while preserving the existing PL/PDF encryption API. The implementation runs entirely in native Oracle PL/SQL on Oracle Database 21c.

Existing applications remain compatible. If an application already calls PLPDF.SETPROTECTION and does not explicitly select an algorithm, PL/PDF continues to use the original RC4 40-bit encryption mode. No application change is required to keep the existing behavior.

For new development, PL/PDF supports four encryption algorithms:

- RC4 40-bit — the original and default encryption mode.
- RC4 128-bit — stronger RC4 encryption for compatibility with older PDF workflows.
- AES 128-bit — modern 128-bit AES encryption.
- AES 256-bit — the recommended modern option when the target PDF viewers support it.

2. Selecting the encryption algorithm

The encryption algorithm is selected with PLPDF_SECURITY.SETALGORITHM after PLPDF.INIT and before PLPDF.SETPROTECTION.

Available constants:

```
plpdf_security.c_algorithm_rc4_40  
plpdf_security.c_algorithm_rc4_128  
plpdf_security.c_algorithm_aes_128  
plpdf_security.c_algorithm_aes_256
```



PL/PDF OffX v5

If SETALGORITHM is not called, RC4 40-bit remains the default. This is intentional for backward compatibility with existing PL/PDF applications.

Example algorithm selection:

```
plpdf_security.setAlgorithm(  
    plpdf_security.c_algorithm_aes_256  
);
```

3. Passwords and permissions

PLPDF.SETPROTECTION continues to be the application-level API for passwords and document permissions.

p_user_pass

The user password is used to open the encrypted document. The user password is optional. If specified, it is required to open the encrypted document. If omitted, the document can be opened without a password, while the configured document permissions still apply.

p_owner_pass

The owner password is used for owner-level access to the protected PDF.

The current permission flags are:

p_print_perm — allow or deny printing.

p_modify_perm — allow or deny document modification.

p_copy_perm — allow or deny copying or extracting document content.

p_annot_forms_perm — allow or deny annotations and form operations.

Viewer applications are responsible for enforcing PDF permissions, so the exact user interface can differ between PDF readers.



4. Backward compatibility

The existing PL/PDF encryption call remains valid without any changes:

```
plpdf.setProtection(  
    p_print_perm    => true,  
    p_modify_perm   => false,  
    p_copy_perm     => false,  
    p_annot_forms_perm => false,  
    p_user_pass     => 'user',  
    p_owner_pass    => 'owner'  
);
```

If `PLPDF_SECURITY.SETALGORITHM` is not called, the document is encrypted with the original RC4 40-bit mode.

This means applications upgraded from earlier PL/PDF versions do not need to be modified unless they want to select RC4 128-bit, AES 128-bit, or AES 256-bit encryption.

5. Recommended algorithm for new applications

AES 256-bit is the preferred option for new applications when compatibility with modern PDF viewers is sufficient.

To use AES 256-bit, select the algorithm before calling `SETPROTECTION`:

```
plpdf_security.setAlgorithm(  
    plpdf_security.c_algorithm_aes_256  
);
```

PL/PDF handles the encryption keys, password processing, random values, encrypted strings, encrypted streams, and PDF output internally. Application code does not need to manage cryptographic keys or initialization vectors.



6. AES-256 example

The following procedure creates a simple AES-256 encrypted PDF. The example allows printing and disables modification, copying, annotations, and form operations.

The user password is user and the owner password is owner.

create or replace procedure test_encrypt_aes256 is

```
l_blob blob;
```

```
begin
```

```
  plpdf.init();
```

```
  -- Select AES-256 encryption.
```

```
  plpdf_security.setAlgorithm(
```

```
    plpdf_security.c_algorithm_aes_256
```

```
  );
```

```
  -- Configure document permissions and passwords.
```

```
  plpdf.setProtection(
```

```
    p_print_perm    => true,
```

```
    p_modify_perm   => false,
```

```
    p_copy_perm     => false,
```

```
    p_annot_forms_perm => false,
```

```
    p_user_pass     => 'user',
```

```
    p_owner_pass    => 'owner'
```

```
  );
```

```
  plpdf.NewPage;
```

```
  plpdf.SetPrintFont('Arial', null, 12);
```

```
  plpdf.PrintCell(100, 20, 'PL/PDF AES-256 encrypted document');
```



PL/PDF OffX v5

```
plpdf.SendDoc(l_blob);  
  
delete from STORE_BLOB;  
  
insert into STORE_BLOB (blob_file, created_date)  
values (l_blob, sysdate);  
  
commit;  
  
end;  
/
```

7. How the example works

PLPDF.INIT starts a new PDF document and resets the encryption state.

PLPDF_SECURITY.SETALGORITHM selects AES 256-bit encryption for the current document.

PLPDF.SETPROTECTION sets the passwords and permission flags.

PLPDF.NEWPAGE creates the first page.

PLPDF.SETPRINTFONT and PLPDF.PRINTCELL create visible document content.

PLPDF.SENDDOC finalizes the PDF and returns the encrypted document as a BLOB.

The example stores the resulting BLOB in STORE_BLOB.

8. Testing the generated PDF

Run the example procedure and save the BLOB from STORE_BLOB as a PDF file.

Open the file in a current PDF viewer such as Adobe Acrobat Reader or Foxit Reader.

Expected behavior:

The viewer asks for a password.

The user password user opens the document.

An incorrect password is rejected.

The text PL/PDF AES-256 encrypted document is visible after the document is opened.

Printing is allowed.

Modification and copying are restricted according to the example settings.



9. Supported algorithms summary

RC4 40-bit

Default mode. Retained for full backward compatibility with existing PL/PDF encryption code.

RC4 128-bit

Available for applications that need stronger RC4 encryption while maintaining compatibility with older PDF environments.

AES 128-bit

Modern AES encryption supported by current PDF viewers.

AES 256-bit

Recommended for new applications where modern PDF viewer compatibility is available. Revision 6 password processing also supports Unicode passwords.