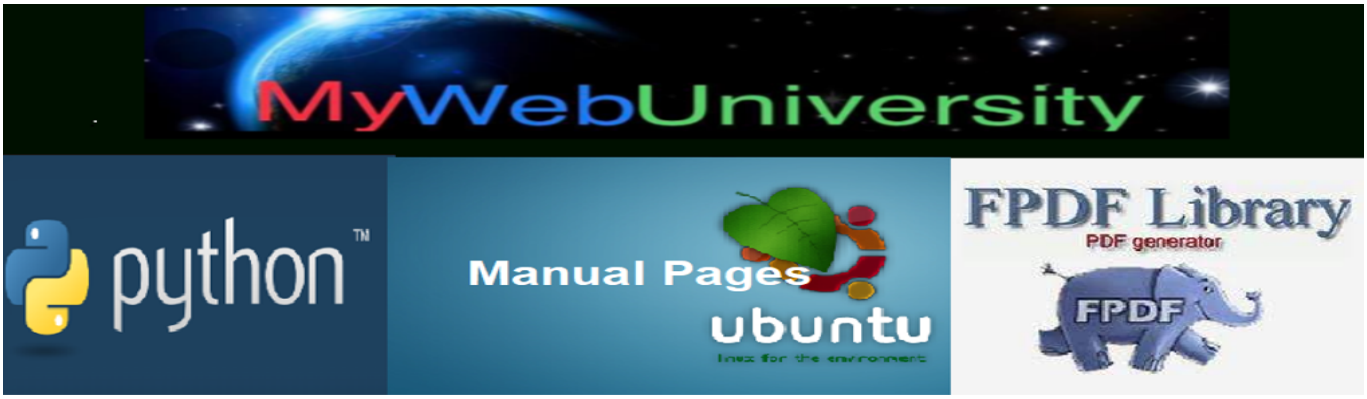

Rocky Enterprise Linux 9.2 Manual Pages on command 'Crypt::AuthEnc::EAX.3pm'

$ man Crypt::AuthEnc::EAX.3pm

Crypt::AuthEnc::EAX(3pm) User Contributed Perl Documentation Crypt::AuthEnc::EAX(3pm)

NAME

Crypt::AuthEnc::EAX - Authenticated encryption in EAX mode

SYNOPSIS

```
### OO interface
use Crypt::AuthEnc::EAX;

# encrypt and authenticate
my $ae = Crypt::AuthEnc::EAX->new("AES", $key, $iv);
$ae->adata_add('additional_authenticated_data1');
$ae->adata_add('additional_authenticated_data2');
my $ct = $ae->encrypt_add('data1');
$ct .= $ae->encrypt_add('data2');
$ct .= $ae->encrypt_add('data3');
my $tag = $ae->encrypt_done();

# decrypt and verify
my $ae = Crypt::AuthEnc::EAX->new("AES", $key, $iv);
$ae->adata_add('additional_authenticated_data1');
$ae->adata_add('additional_authenticated_data2');
my $pt = $ae->decrypt_add('ciphertext1');
```

```
$pt .= $ae->decrypt_add('ciphertext2');
$pt .= $ae->decrypt_add('ciphertext3');
my $tag = $ae->decrypt_done();
die "decrypt failed" unless $tag eq $expected_tag;

#or
my $result = $ae->decrypt_done($expected_tag); # 0 or 1

### functional interface
use Crypt::AuthEnc::EAX qw(eax_encrypt_authenticate eax_decrypt_verify);

my ($ciphertext, $tag) = eax_encrypt_authenticate('AES', $key, $iv, $adata, $plaintext);
my $plaintext = eax_decrypt_verify('AES', $key, $iv, $adata, $ciphertext, $tag);
```

DESCRIPTION

EAX is a mode that requires a cipher, CTR and OMAC support and provides encryption and authentication. It is initialized with a random IV that can be shared publicly, additional authenticated data which can be fixed and public, and a random secret symmetric key.

EXPORT

Nothing is exported by default.

You can export selected functions:

```
use Crypt::AuthEnc::EAX qw(eax_encrypt_authenticate eax_decrypt_verify);
```

FUNCTIONS

eax_encrypt_authenticate

```
my ($ciphertext, $tag) = eax_encrypt_authenticate($cipher, $key, $iv, $adata, $plaintext);

# $cipher .. 'AES' or name of any other cipher with 16-byte block len
# $key ..... AES key of proper length (128/192/256bits)
```

```
    # $iv ..... unique initialization vector (no need to keep it secret)
    # $adata ... additional authenticated data

  eax_decrypt_verify
    my $plaintext = eax_decrypt_verify($cipher, $key, $iv, $adata, $ciphertext, $tag);
    # on error returns undef

METHODS
  new
    my $ae = Crypt::AuthEnc::EAX->new($cipher, $key, $iv);
    #or
    my $ae = Crypt::AuthEnc::EAX->new($cipher, $key, $iv, $adata);

    # $cipher .. 'AES' or name of any other cipher with 16-byte block len
    # $key ..... AES key of proper length (128/192/256bits)
    # $iv ..... unique initialization vector (no need to keep it secret)
    # $adata ... additional authenticated data (optional)

  adata_add
    $ae->adata_add($adata);                  # can be called multiple times

  encrypt_add
    $ciphertext = $ae->encrypt_add($data);       # can be called multiple times

  encrypt_done
    $tag = $ae->encrypt_done();                  # returns $tag value

  decrypt_add
    $plaintext = $ae->decrypt_add($ciphertext);    # can be called multiple times

  decrypt_done
    my $tag = $ae->decrypt_done;          # returns $tag value
    #or
```

```
my $result = $ae->decrypt_done($tag);  # returns 1 (success) or 0 (failure)
```

clone

```
my $ae_new = $ae->clone;
```

SEE ALSO

- CryptX, Crypt::AuthEnc::CCM, Crypt::AuthEnc::GCM, Crypt::AuthEnc::OCB

- <https://en.wikipedia.org/wiki/EAX_mode>