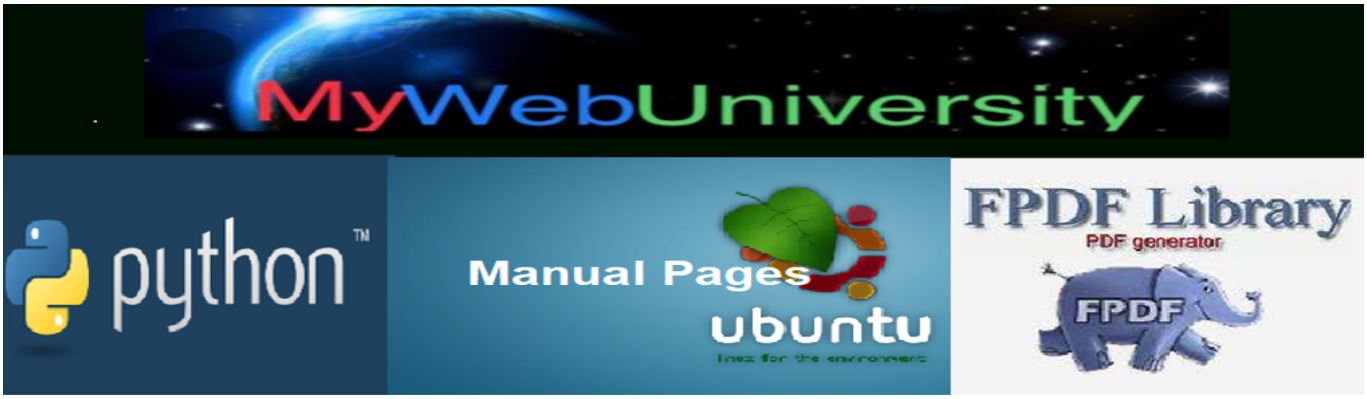

Rocky Enterprise Linux 9.2 Manual Pages on command 'Crypt::AuthEnc::ChaCha20Poly1305.3pm'

$ man Crypt::AuthEnc::ChaCha20Poly1305.3pm

Crypt::AuthEnc::ChaCha20Poly130UsermContributed Perl DocumenCrypt::AuthEnc::ChaCha20Poly1305(3pm)

NAME

Crypt::AuthEnc::ChaCha20Poly1305 - Authenticated encryption in ChaCha20-Poly1305 mode

SYNOPSIS

```
### OO interface
use Crypt::AuthEnc::ChaCha20Poly1305;

# encrypt and authenticate
my $ae = Crypt::AuthEnc::ChaCha20Poly1305->new($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 $ae = Crypt::AuthEnc::ChaCha20Poly1305->new($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::ChaCha20Poly1305 qw(chacha20poly1305_encrypt_authenticate chacha20poly1305_decrypt_verify);

my ($ciphertext, $tag) = chacha20poly1305_encrypt_authenticate($key, $iv, $adata, $plaintext);
my $plaintext = chacha20poly1305_decrypt_verify($key, $iv, $adata, $ciphertext, $tag);
```

DESCRIPTION

Provides encryption and authentication based on ChaCha20 + Poly1305 as defined in RFC 7539 - <https://tools.ietf.org/html/rfc7539>

EXPORT

Nothing is exported by default.

You can export selected functions:

```
use Crypt::AuthEnc::ChaCha20Poly1305 qw(chacha20poly1305_encrypt_authenticate chacha20poly1305_decrypt_verify);
```

FUNCTIONS

chacha20poly1305_encrypt_authenticate

```
my ($ciphertext, $tag) = chacha20poly1305_encrypt_authenticate($key, $iv, $adata, $plaintext);

# $key ..... key of proper length (128 or 256 bits / 16 or 32 bytes)
# $iv ..... initialization vector (64 or 96 bits / 8 or 12 bytes)
```

```
    # $adata ... additional authenticated data (optional)

  chacha20poly1305_decrypt_verify
    my $plaintext = chacha20poly1305_decrypt_verify($key, $iv, $adata, $ciphertext, $tag);
    # on error returns undef

METHODS
  new
    my $ae = Crypt::AuthEnc::ChaCha20Poly1305->new($key, $iv);

    # $key ..... encryption key of proper length (128 or 256 bits / 16 or 32 bytes)
    # $iv ..... initialization vector (64 or 96 bits / 8 or 12 bytes)

  adata_add
    Add additional authenticated data.  Can be called before the first "encrypt_add" or
    "decrypt_add";

    $ae->adata_add($aad_data);                  # 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)
```

set_iv

my $ae = Crypt::AuthEnc::ChaCha20Poly1305->new($key)->set_iv($iv);

$iv ..... initialization vector (64 or 96 bits / 8 or 12 bytes)

set_iv_rfc7905

See <https://tools.ietf.org/html/rfc7905>

my $ae = Crypt::AuthEnc::ChaCha20Poly1305->new($key)->set_iv_rfc7905($iv, $seqnum);

$iv ..... initialization vector (96 bits / 12 bytes)

$seqnum .. 64bit integer (sequence number)

clone

my $ae_new = $ae->clone;

SEE ALSO

? CryptX, Crypt::AuthEnc::GCM, Crypt::AuthEnc::CCM, Crypt::AuthEnc::EAX, Crypt::AuthEnc::OCB

? <https://tools.ietf.org/html/rfc7539>