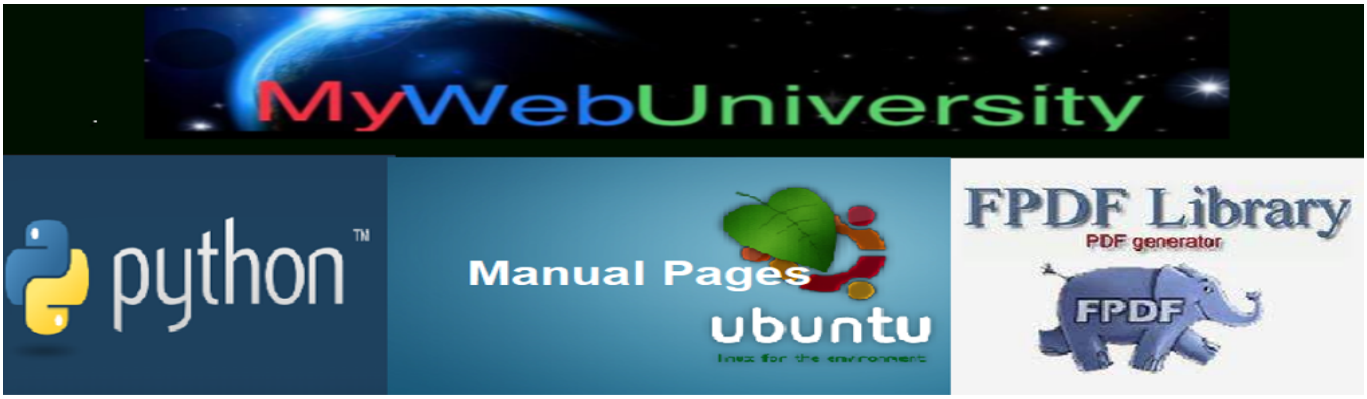

Rocky Enterprise Linux 9.2 Manual Pages on command 'Crypt::AuthEnc::GCM.3pm'

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

NAME

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

SYNOPSIS

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

# encrypt and authenticate
my $ae = Crypt::AuthEnc::GCM->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::GCM->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::GCM qw(gcm_encrypt_authenticate gcm_decrypt_verify);

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

DESCRIPTION

Galois/Counter Mode (GCM) - provides encryption and authentication.

EXPORT

Nothing is exported by default.

You can export selected functions:

```
use Crypt::AuthEnc::GCM qw(gcm_encrypt_authenticate gcm_decrypt_verify);
```

FUNCTIONS

gcm_encrypt_authenticate

```
my ($ciphertext, $tag) = gcm_encrypt_authenticate($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 ..... initialization vector
# $adata ... additional authenticated data
```

gcm_decrypt_verify

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

METHODS

new

```
my $ae = Crypt::AuthEnc::GCM->new($cipher, $key);
#or
my $ae = Crypt::AuthEnc::GCM->new($cipher, $key, $iv);

# $cipher .. 'AES' or name of any other cipher
# $key ..... encryption key of proper length
# $iv ..... initialization vector (optional, you can set it later via iv_add method)
```

iv_add

Set initialization vector (IV).

```
$ae->iv_add($iv_data);                    #can be called multiple times
```

adata_add

Add additional authenticated data. Can be called after all "iv_add" calls but 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)
```

reset

```
$ae->reset;
```

clone

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

SEE ALSO

- CryptX, Crypt::AuthEnc::CCM, Crypt::AuthEnc::EAX, Crypt::AuthEnc::OCB
- <https://en.wikipedia.org/wiki/Galois/Counter_Mode>