OpenSSL AES_GCM

Michele La Manna
Dept. of Information Engineering
University of Pisa
michele.lamanna@phd.unipi.it
Version: 2022-04-05

Exercise

AES_GCM()

OpenSSL (hands-on)

Write a single file which encrypts and decrypts a message using AES_gcm()

Tip: Write a main and two "utility functions", Encrypt and Decrypt.

Use hard-coded message, key and IV.

45 minutes

Checkpoint (1/3): Main

UNIVERSITÀ DI PISA

```
93  int main (void)
94  {
95      unsigned char msg[] = "Short message";
96      //create key
97      unsigned char key_gcm[]="1234567890123456";
98      unsigned char iv_gcm[]= "123456780912";
99      unsigned char *cphr_buf;
100     unsigned char *tag_buf;
101     int cphr_len;
102     int tag_len;
103     int pt_len = sizeof(msg);
104     cphr_buf=(unsigned char*)malloc(pt_len+16);
105     tag_buf=(unsigned char*)malloc(16);
106     gcm_encrypt(msg, pt_len, iv_gcm, 12, key_gcm, iv_gcm, 12, cphr_buf, tag_buf);
107     cout<<"CT:"<<endl;
108     BIO_dump_fp (stdout, (const char *)cphr_buf, pt_len);
109     cout<<"Tag:"<<endl;
110     BIO_dump_fp (stdout, (const char *)tag_buf, 16);
111     unsigned char *dec_buf;
112     dec_buf=(unsigned char*)malloc(pt_len);
113     gcm_decrypt(cphr_buf, pt_len, iv_gcm, 12, tag_buf, key_gcm, iv_gcm, 12, dec_buf);
114     cout<<"PT:"<<endl;
115     BIO_dump_fp (stdout, (const char *)dec_buf, pt_len);
116     return 0;
117 }
```

Checkpoint (2/3): Encrypt

UNIVERSITÀ DI PISA

```
15  int gcm_encrypt(unsigned char *plaintext, int plaintext_len,
16                  unsigned char *aad, int aad_len,
17                  unsigned char *key,
18                  unsigned char *iv, int iv_len,
19                  unsigned char *ciphertext,
20                  unsigned char *tag)
21  {
22      EVP_CIPHER_CTX *ctx;
23      int len;
24      int ciphertext_len;
25      // Create and initialise the context
26      if(!(ctx = EVP_CIPHER_CTX_new()))
27          handleErrors();
28      // Initialise the encryption operation.
29      if(1 != EVP_EncryptInit(ctx, EVP_aes_128_gcm(), key, iv))
30          handleErrors();
31
32      //Provide any AAD data. This can be called zero or more times as required
33      if(1 != EVP_EncryptUpdate(ctx, NULL, &len, aad, aad_len))
34          handleErrors();
35
36      if(1 != EVP_EncryptUpdate(ctx, ciphertext, &len, plaintext, plaintext_len))
37          handleErrors();
38      ciphertext_len = len;
39      //Finalize Encryption
40      if(1 != EVP_EncryptFinal(ctx, ciphertext + len, &len))
41          handleErrors();
42      ciphertext_len += len;
43      /* Get the tag */
44      if(1 != EVP_CIPHER_CTX_ctrl(ctx, EVP_CTRL_AEAD_GET_TAG, 16, tag))
45          handleErrors();
46      /* Clean up */
47      EVP_CIPHER_CTX_free(ctx);
48      return ciphertext_len;
49  }
```

Checkpoint (3/3): Decrypt


```c
int gcm_decrypt(unsigned char *ciphertext, int ciphertext_len,
                unsigned char *aad, int aad_len,
                unsigned char *tag,
                unsigned char *key,
                unsigned char *iv, int iv_len,
                unsigned char *plaintext)
{
    EVP_CIPHER_CTX *ctx;
    int len;
    int plaintext_len;
    int ret;
    /* Create and initialise the context */
    if(!(ctx = EVP_CIPHER_CTX_new()))
        handleErrors();
    if(!EVP_DecryptInit(ctx, EVP_aes_128_gcm(), key, iv))
        handleErrors();
    //Provide any AAD data.
    if(!EVP_DecryptUpdate(ctx, NULL, &len, aad, aad_len))
        handleErrors();
    //Provide the message to be decrypted, and obtain the plaintext output.
    if(!EVP_DecryptUpdate(ctx, plaintext, &len, ciphertext, ciphertext_len))
        handleErrors();
    plaintext_len = len;
    /* Set expected tag value. Works in OpenSSL 1.0.1d and later */
    if(!EVP_CIPHER_CTX_ctrl(ctx, EVP_CTRL_AEAD_SET_TAG, 16, tag))
        handleErrors();
    /*
     * Finalise the decryption. A positive return value indicates success,
     * anything else is a failure - the plaintext is not trustworthy.
     */
    ret = EVP_DecryptFinal(ctx, plaintext + len, &len);
    /* Clean up */
    EVP_CIPHER_CTX_cleanup(ctx);
    if(ret > 0) {
        /* Success */
        plaintext_len += len;
        return plaintext_len;
    } else {
        /* Verify failed */
        return -1;
    }
```