OpenSSL: symmetric encryption

CATALDO BASILE

< CATALDO.BASILE@ POLITO.IT >

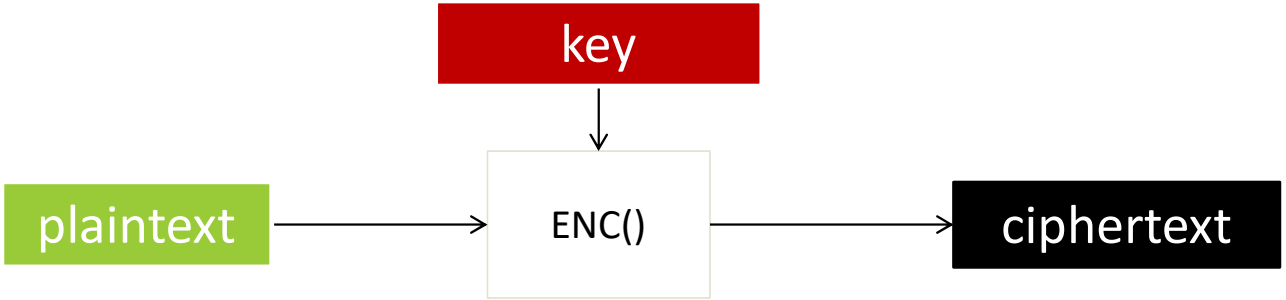
POLITECNICO DI TORINO

Agenda

introduction to the main OpenSSL characteristics

programs in C

Symmetric encryption



used to enforce confidentiality

- only who knows a secret (e.g., a key) can perform some operations

main algorithm design: block vs. stream

- AES vs. ChaCha20

additional decisions for block algorithms

- padding yes/no (the standard is PKCS#5)
- modes of operations: ECB, CBC, OFB, CFB, …

Symmetric encryption in OpenSSL

symmetric encryption works (or can be adapted to work) on a limited number of bytes at the time

- e.g., block algorithms
  - by definition
- e.g., stream algorithms generate sequences of bytes (keystream)
  - but they can be continued if we store the states of the keystream generator

OpenSSL implements symmetric encryption with incremental functions

- the encryption context manages the incremental updates

Incremental symmetric encryption

Encryption (pseudo-code):

```
ctx = context_initialize(encrypt, cipher, mode, key, iv, …);
cycle:
    ciphertext_fragment = encrypt_update(ctx, plaintext_fragment);
end:
ciphertext_fragment = encrypt_finalize(ctx);
```

Decryption:

```
ctx = context_initialize(decrypt, cipher, mode, key, iv, …);
cycle:
    plaintext_fragment = decrypt_update(ctx, ciphertext_fragment);
end:
plaintext_fragment = decrypt_finalize(ctx);
```

Incremental encryption in Openssl

done with the EVP API

- *openssl/evp.h*
- a unique interface for all symmetric encryption algorithms
  - https://www.openssl.org/docs/man1.1.1/man3/EVP_EncryptInit.html
  - https://www.openssl.org/docs/man3.0/man3/EVP_EncryptInit.html

EVP API provides

- a data structure (the context): *EVP_CIPHER_CTX*
- functions for:
  - context creation/destruction: *EVP_CIPHER_CTX_new /* EVP_CIPHER_CTX_cleanup / …
  - context initialization: e.g., (*EVP_EncryptInit* / *EVP_DecryptInit)* or *EVP_CipherInit*
  - encryption/decryption: *EVP_EncryptUpdate* or *EVP_CipherUpdate*
  - finalization: *EVP_EncryptFinal* or *EVP_CipherFinal*

Ciphers in Openssl

symmetric algorithms are “objects” to be loaded in the context

- to use an algorithm, obtain a pointer to the proper sym-enc object
  - plug it into the context
- the EVP provides means to get these pointers
  - e.g., to have a reference to a Blowfish-CBC object you can use:

```
EVP_CIPHER *c = EVP_bf_cbc();
```

  - HINT: names look like the ones you use in OpenSSL in command line
    - e.g. -aes-128-cbc → EVP_aes_128_cbc
    - start with the ”EVP_” prefix then substitute s/”-”/”_”