Praktyczne aspekty wykorzystania nowoczesnej kryptografii

Wojciech A. Koszek
<dunstan@freebsd.czest.pl>

Wprowadzenie

- Kryptologia
- Nauka dotycząca przekazywania danych w poufny sposób. W jej skład wchodzi kryptografia i kryptoanaliza

Kryptografia

- Zajmuje się tworzeniem metod szyfrowania.
- Głównymi zagadnieniami kryptografii są:
  - algorytmy kryptograficzne
  - protokoły kryptograficzne
  - funkcje skrótu

Algorytmy kryptograficzne

- Opisują sposób zamiany bloków tekstu jawnego
- Algorytmy dzielimy na
  - symetryczne
  - asymetryczne

Algorytmy symetryczne

- Algorytmy, w krótych szyfrowanie i deszyfrowanie odbywa się przy pomocy tego samego klucza.
- Przykłady:
  - DES (Lucifer)
  - AES (Rijandael)

Algorytmy asymetryczne

- Algorytmy asymetryczne to takie, w których klucz szyfrujący jest inny niż klucz deszyfrujący.
- Przykłady:
  - RSA

Funkcje skrótu

- Funkcje przyjmujące pewną ilość danych, oraz zwracające unikalny ciąg identyfikujący te dane.
- Przykładami funkcji skrótu mogą być
  - SHA-1
  - MD5
  - RMD-160

Funkcje skrótu (II)

- Wartości poszczególnych funkcji dla ciągu “kryptografia”:

MD5	d29d2fd37d23dedfefcc3c6a41c42410
SHA-1	254f0eb08eaf1d33e0988d64f861c8681a7e6579
RMD-160	4e92978a3a3da55a8fcb2260cee89ebbcc28d058

Protokoły kryptograficzne

- Protokoły kryptograficzne mają na celu:
  - przesyłanie pewnej porcji danych w poufny sposób
  - potwierdzenie tożsamości nadawcy i odbiorcy
  - zapewnienie o pochodzeniu pewnej porcji danych (potwierdzenie elektronicznego podpisu)

Protokoły kryptograficzne: SSL

Protokół SSL *(Secure Socket Layer)*

- Protokół SSL jest przykładem protokołu:
  - zapewniającego autentyczność
  - gwarantującego pufność danych
  - gwarantującego integralność

Protokół SSL (II)

- Protokół SSL pełni funkcję pośrednika między warstwą TCP/IP a protokołami użytkowymi.
- Możliwe jest wykorzystanie protoku SSL do zabezpieczania wielu popularnych usług (HTTP, POP3, IMAP)

OpenSSL: przykłady

OpenSSL

- Zalety biblioteki:
  - jest wolnodostępna
  - jest przenośna
  - jest rozszerzalna
- Wady:
  - błędy narażające bezpieczeństwo usług korzystających z biblioteki

OpenSSL: szyfrowanie symetryczne

- Proces szyfrowania danych przy pomocy algorytmu DES zawartych w pliku *dane.txt* należy przeprowadzić poprzez:

```
openssl enc -e -des-cbc < dane.txt > tajne.txt
```

OpenSSL: szyfrowanie asymetryczne

- Pierwszy krok to generacja klucza:

```
openssl genrsa -out key 1024
```

- *Następnie należy uzyskać publiczną część powstałego klucza:*

```
openssl rsa -in key -pubout -out key.pub
```

OpenSSL: szyfrowanie asymetryczne (II)

- Szyfrowanie kluczem publicznym:

```
openssl rsautl -encrypt -pubin -inkey key.pub -in dane.txt -out tajne.txt
```

- Szyfrowanie kluczem prywatnym:

```
openssl rsautl -encrypt -inkey key -in dane.txt -out tajne.txt
```

Wykorzystanie biblioteki OpenSSL

- Podstawy:
  - Struktury i funkcje *BIO:*
    - Odpowiedniki standardowych funkcji biblioteki *libc* operujących na strukturach *FILE*
  - struktury i funkcje *EVP_**:
    - określenie rodzaju algorytmu
    - określenie funkcji skrótu
    - określanie parametrów procesu szyfrowania

Struktury *BIO* (I)

- Podstawowe operacje:
  - *BIO* BIO_new(..)* - tworzenie struktury
  - *BIO* BIO_set_fp(..)* - łączenie struktury BIO ze strukturą FILE
  - *BIO* BIO_push(..)* - możliwość wiązania struktur, czego wynikiem jest łańcuch przepływu danych.

Struktury *BIO* (II)

```
BIO *sd_o;
FILE *out;
[..]
sd_o = BIO_new(BIO_s_file());
BIO_set_fp(sd_o, out, BIO_CLOSE);
```

Struktury *EVP_** (I)

- Podstawowe operacje:
  - Inicjalizacja szyfru – przykładowo *EVP_des_cbc()*
  - Inicjalizacja wartości klucza oraz wektora początkowego używanego podczas szyfrowania – funkcja *EVP_BytesToKey(..)*

Struktury *EVP_** (II)

```
const EVP_CIPHER *c;
const EVP_MD *md;
[..]
c = EVP_des_cbc();
md = EVP_sha1();
```

Struktury *EVP_** (III)

Przygotowanie klucza zgodnego z *PKCS#5*:

```
unsigned char salt[PKCS5_SALT_LEN],
unsigned char key[EVP_MAX_KEY_LENGTH];
unsigned char iv[EVP_MAX_IV_LENGTH];

EVP_BytesToKey(c, md, salt, buf, strlen
  (buf), 1, key, iv);
```

Wartości losowe

Biblioteka OpenSSL zapewnia interfejs do łatwego generowania losowego ciągu bitów, w tym przypadku potrzebnego do uzyskania tzw. soli:

```
RAND_bytes(salt, sizeof(salt));
```

Tworzenie łańcucha szyrfującego

```
BIO *cipher_bio;
cipher_bio = BIO_new(BIO_f_cipher());
BIO_set_cipher(cipher_bio, c, key, iv, 1);
[..]
bio = BIO_push(cipher_bio, sd_in);
```

Proces szyfrowania

- Dzięki wykorzystaniu struktur BIO proces szyfrowania sprowadza się do wykonania pętli:

```
while ((l = BIO_read(bio, buf, sizeof(buf))) > 0)
{
    BIO_write(sd_out, buf, l);
}
```

Zakończenie działania

- Zwolnienie zasobów przy pomocy *void BIO_free(..)*
- Wywołanie funkcji **_cleanup*

Podsumowanie

- Przyszłość i rozwój kryptografii
  - kryptografia kwantowa
  - kryptografia bazująca na krzywych eliptycznych

Koniec

Dziękuję i zapraszam do zadawania pytań