Podstawy kryptografii

Jednokierunkowe funkcje skrótu

© copyright by Cyberskiller Sp. z o.o., 2020.
Wszystkie prawa zastrzeżone.
Wszelkie zmiany, kopiowanie i wykorzystywanie
dozwolone wyłącznie za uprzednią zgodą Cyberskiller Sp. z o.o.

email: contact@cyberskiller.com
cyberskiller.com

Podstawy kryptografii

Jednokierunkowe funkcje skrótu

Spis Treści

© copyright by Cyberskiller Sp. z o.o., 2020.
Wszystkie prawa zastrzeżone.
Wszelkie zmiany, kopiowanie i wykorzystywanie
dozwolone wyłącznie za uprzednią zgodą Cyberskiller Sp. z o.o.

email: contact@cyberskiller.com
cyberskiller.com

Wiadomości podstawowe

Jednokierunkowe funkcje skrótu zwane zazwyczaj funkcjami hashującymi to algorytmy, które dla dowolnej ilości danych wejściowych generują ciąg binarny o określonej, stałej długości. Ciąg ten zapisywany jest najczęściej w postaci szesnastkowej. Choć długość tego ciągu może być dowolna (specyficzna dla danego algorytmu) to, z powodów praktycznych, wykorzystuje się algorytmy generujące ciągi od 128 do 512 bitów. Przekłada się to na wartości szesnastkowe o długościach od 32 do 128 znaków. Popularne algorytmy jednokierunkowych funkcji skrótu to np.: **`md5`**, **`sha-1`** lub **`sha-2`**.

Przykłady:
`MD5("cyberskiller") = 8c6abdfb40a76a9936d37fa47a4cae93`
`SHA256("cyberskiller") =`
`fc06e1bd631ae489e5fdaf6f143231e7d5195483a4317fd34aab02197ff07a9f`

Ważną cechą jednokierunkowych funkcji skrótu jest ich duża "wrażliwość" na nawet niewielką zmianę danych wejściowych. Wyliczmy wartość funkcji **`sha256`** dla ciągu bitów
`001001101010010:`
`SHA256("001001101010010") =`
`871f55bda6963874621809faf300f969a40bee38808ecc286c596d67929e9820`

Jeśli dokonamy zmiany wartości choćby jednego bitu (pierwszego), uzyskamy całkiem inną (istotnie różną) wartość:
`SHA256("101001101010010") =`
`cf1d505c0e4311c76f9a8ff4bddfdbe11e7d49e1a7f46bbe07266c9d9a8ebbc`

Właściwość ta ma duże znaczenie jeśli funkcje skrótu wykorzystywane są do generowania podpisów lub weryfikowania integralności danych. Załóżmy, że mamy do czynienia z wiadomością o następującej treści:

"Koszty operacji finansowych, które muszą zostać poniesione szacuje się na 1250,34 złote"

Aby mieć pewność, że treść wiadomości nie została zmieniona dostarczono nam także innym kanałem wartość funkcji **`sha256`** dla tej wiadomości:
`e1a15f833f1e10245311637889c59eef41a8697bd043c793117ce92c5b725e20.`

Jeśli dokonamy weryfikacji, to znaczy samodzielnie wygenerujemy skrót wiadomości dostaniemy wartość:
`8bc03142891772bd8a36a23498c8004d1f3948cab55945be6d1155c0938e6a9b`

Jak widać wartości skrótów nie są takie same. Oznacza to, że treść wiadomości nie jest tożsama z tą, dla której wygenerowano przesłaną nam wartość skrótu. Choć nie jesteśmy w stanie ustalić jaka powinna być poprawna treść, mamy pewność, że to co otrzymaliśmy nie jest dokładnie tym co zostało wysłane.
Prawdziwa treść wiadomości to:

"Koszty operacji finansowych, które muszą zostać poniesione
szacuje się na 125,34złote"

© copyright by Cyberskiller Sp. z o.o., 2020.
Wszystkie prawa zastrzeżone.
Wszelkie zmiany, kopiowanie i wykorzystywanie
dozwolone wyłącznie za uprzednią zgodą Cyberskiller Sp. z o.o.

Jak wspomniano wcześniej rozmiar danych wejściowych dla algorytmu funkcji skrótu może być dowolny, zaś wynik posiada zawsze tę samą długość. Oznacza to, że przy nieskończonej ilości możliwych ciągów wejściowych uzyskać możemy jedynie 2^n wartości wyjściowych o długości **n** bitów. Oznacza to, że jeśli przygotujemy zestaw 2^n+1 różnych danych wejściowych to przynajmniej dla dwóch z nich otrzymamy dokładnie tę samą wartość funkcji skrótu. Sytuacja ta nazywa się kolizją i choć jak widać jest nieunikniona to jest wysoce niepożądana.

Wyliczanie wartości funkcji skrótu

W systemach Linux/UNIX mamy do dyspozycji kilka programów, które wyliczają wartości funkcji skrótu z wykorzystaniem wybranych algorytmów. Najpopularniejsze z nich to: **md-5sum**, **shasum**, **sha1sum**, **sha256sum**, itd.

Przed wykorzystaniem któregokolwiek programu warto zajrzeć do podręcznika systemowego. Jak czytamy w podręczniku systemowym, np. program shasum wypisuje lub sprawdza sumy kontrolne, co oznacza, że w istocie wylicza on wartości funkcji skrótu dla danych wejściowych wprowadzanych ze standardowego wejścia lub pliku.

```
man shasum

NAME
       shasum - Print or Check SHA Checksums

SYNOPSIS
        Usage: shasum [OPTION]... [FILE]...
        Print or check SHA checksums.
        With no FILE, or when FILE is -, read standard input.

          -a, --algorithm   1 (default), 224, 256, 384, 512, 512224, 512256
          -b, --binary      read in binary mode
          -c, --check       read SHA sums from the FILEs and check them
              --tag         create a BSD-style checksum
          -t, --text        read in text mode (default)
          -U, --UNIVERSAL   read in Universal Newlines mode
                                produces same digest on Windows/Unix/Mac
          -0, --01          read in BITS mode
                                ASCII ‚0’ interpreted as 0-bit,
                                ASCII ‚1’ interpreted as 1-bit,
                                all other characters ignored
```

Jak widać do dyspozycji mamy kilka algorytmów generujących skróty różnej długości. Wyliczmy skrót o długości 256 bitów dla ciągu “Ala ma kota”. Można to zrobić w następujący sposób:

```
echo -n "Ala ma kota." | shasum -a 256
```

W wyniku dostajemy:

```
9b9ae722342758f4bb6a5dd94470ddfab3c95cda15186231f53bb4666e657605  -
```


© copyright by Cyberskiller Sp. z o.o., 2020.
Wszystkie prawa zastrzeżone.
Wszelkie zmiany, kopiowanie i wykorzystywanie
dozwolone wyłącznie za uprzednią zgodą Cyberskiller Sp. z o.o.

email: contact@cyberskiller.com
cyberskiller.com

Poniżej znajduje się krótki przegląd algorytmów zaimplementowanych w programie **shasum**.

- algorytm **sha1**, skrót 160 bitów:

```
echo -n "Ala ma kota." | shasum -a 1
3808b409a7c16ff3ac19059844e347505dc81369
```

- algorytm **sha2**, skróty 224, 256, 384 i 512 bitów:

```
echo -n "Ala ma kota." | shasum -a 224
4acbb6e9552ee824e29331cf48e4b0286b54a2a2d129383d052d6f8b
echo -n "Ala ma kota." | shasum -a 256
9b9ae722342758f4bb6a5dd94470ddfab3c95cda15186231f53bb4666e657605
echo -n "Ala ma kota." | shasum -a 384
76c05f5a57e79f825c41c70d9f91148b50ef1cfb91dcad86b57ff29e9a1e67a9 \
058f0ef6f3bfe502796cf97afb5c87e7
echo -n "Ala ma kota." | shasum -a 512
d033c1d39e606bb7292338a68563cb4f8c45e0de842d554c3df40e446507a2a1 \
6bd924a98228071bbbc4144bc318f0711aa1154fc132ad42bff7b0590c31e461
```

Ponadto istnieje jeszcze program sha1sum, który wylicza skrót o długości 160 bitów wykorzystując algorytm SHA1.

```
echo -n "Ala ma kota." | sha1sum
3808b409a7c16ff3ac19059844e347505dc81369
```

Algorytm SHA3 nie jest bezpośrednio dostępny w systemach Linux/UNIX w postaci oddzielnego programu. Można go znaleźć w bibliotece OpenSSL, lub jako metodę zaimplementowaną w bibliotece hashlib w Pythonie3.

- Wyliczanie skrótu SHA3 za pomocą biblioteki OpenSSL:

```
echo -n "Ala ma kota." | openssl dgst -sha3-256
(stdin)= cd8133f8c4d06eaa7209ec943f7deb0ec1bf985a0dd4b4a71d7a3eca-
      6ca6e012
```

- wyliczanie skrótu SHA3 z wykorzystaniem biblioteki **hashlib** z Pythona3:

```
import hashlib
hash = hashlib.sha3_256()
hash.update(b"Ala ma kota.")
hash.hexdigest()
```

  Wynik:

```
Out[4]: 'cd8133f8c4d06eaa7209ec943f7deb0ec1bf985a0dd4b4a71d7a3eca-
      6ca6e012'
```

Aby wyliczyć skrót MD5 można wykorzystać program **md5sum**:

```
echo -n "Ala ma kota." | md5sum
db7d23e3f08230238cda3303a4f3a79a  -
```


© copyright by Cyberskiller Sp. z o.o., 2020.
Wszystkie prawa zastrzeżone.
Wszelkie zmiany, kopiowanie i wykorzystywanie
dozwolone wyłącznie za uprzednią zgodą Cyberskiller Sp. z o.o.

Z uwagi na udowodnione słabości współcześnie raczej nie korzysta się już z funkcji MD5.

Zwróćmy uwagę na parametr **-n** pojawiający się podczas wywołania polecenia **echo**. Jego obecność powoduje, że do ciągu w cudzysłowach nie jest dodawany znak końca linii. Jego obecność ma kolosalny wpływ na wyliczaną wartość. Warto to sprawdzić, porównując wyniki wykonania poniższych poleceń:

```
echo -n "Ala ma kota." | shasum -a 256
echo "Ala ma kota." | shasum -a 256
```

Możemy także sprawdzić długość ciągu znaków jaki wypisuje na standardowe wyjście komanda **echo**:

```
echo -n "Ala ma kota." | wc -c
$ 12
echo "Ala ma kota." | wc -c
$ 13
```

Jak widać, w drugim przypadku, liczba bajtów, która trafia na standardowe wyjście jest o jeden większa. Ten dodatkowy bajt to znak końca linii. Jeśli kolejny program w potoku to **shasum**, wówczas wygeneruje on skrót dla ciągu zawierającego ten dodatkowy bajt. W przypadku generowania skrótów powinniśmy mieć pełną świadomość dla jakich danych te skróty są generowane.

Komenda **shasum** może generować także skróty (sumy kontrolne) dla plików. Popatrzmy na poniższy przykład:

```
echo "Ala ma kota" > plik.txt
shasum -a 256 plik.txt
```

Sprawdźmy jeszcze, jaką długość ma **plik.txt**:

```
cat plik.txt | wc -c
$ 13
```

Chociaż zdanie zawiera 12 znaków, podobnie jak poprzednio, na końcu pojawia się znak końca linii, który traktowany jest jako element danych wejściowych dla programu **shasum**. Aby tego uniknąć należy do komendy echo dodać parametr **-n** do wywołania.

© copyright by Cyberskiller Sp. z o.o., 2020.
Wszystkie prawa zastrzeżone.
Wszelkie zmiany, kopiowanie i wykorzystywanie
dozwolone wyłącznie za uprzednią zgodą Cyberskiller Sp. z o.o.

Kod uwierzytelniania wiadomości

Kod uwierzytelniana wiadomości (Message Authentication Code) to jednokierunkowa funkcja skrótu wykorzystująca klucz tajny do wygenerowania skrótu wiadomości. Obecność tajnego klucza stanowi w tym przypadku dodatkowe zabezpieczenie gwarantujące oprócz integralności także możliwość uwierzytelnienia. Poprawność wiadomości mogą w tym przypadku sprawdzić jedynie osoby lub procesy posiadające wspomniany tajny klucz.

HMAC (Keyed-Hash Message Authentication Code albo Hash-based Message Authentication Code) to szczególny rodzaj kodu uwierzytelniania wiadomości (MAC), w którym wykorzystuje się kryptograficzną funkcję skrótu i tajny klucz kryptograficzny.

Aby wygenerować kod HMAC dla określonych danych wejściowych (wykorzystując funkcję skrótu SHA2-256) można skorzystać z biblioteki **`openssl`**:

```
echo -n "Ala ma kota." | openssl dgst -hmac caf6d31c3aff
(stdin)= f287a92525146afce29ab433505228f55bb8cd249355045a031c4c77d0b7262c
```

gdzie ciąg **`caf6d31c3aff`** jest tajnym kluczem. Jeśli chcielibyśmy wykorzystać algorytm SHA2-512, to należy dodać do poprzedniego wywołania odpowiedni parametr:

```
echo -n "Ala ma kota." | openssl dgst -sha512 -hmac caf6d31c3aff
(stdin)= 3c7d8fa913b068eb388243e6c314ae1c79020a1ee78fd2a5a6f0ae67a4c \
444ec06ec4de4cfd3e306f9c5141e670a758205e173bd3c9c4694ab5bdfeaf180b6c0
```

Atak słownikowy na funkcje skrótu

Jak wskazuje nazwa “jednokierunkowa funkcja skrótu” - nie ma możliwości odtworzenia danych wejściowych na podstawie informacji zawartej w wygenerowanym skrócie. Pomimo to, dysponując pewną wiedzą, można podjąć próbę odgadnięcia z jakich danych wejściowych wygenerowany został skrót. Przyjmijmy, że posiadamy skrót dla ciągu wejściowego składającego się z ośmiu małych liter alfabetu łacińskiego. Liczba wszystkich możliwych ciągów tego typu wynosi $26^8 = 208\,827\,064\,576$. Można wygenerować listę zawierającą wszystkie te ciągi i obliczać skróty dla poszczególnych z nich porównując je z posiadanym wzorcem. Taka lista, zawierająca pewien pokaźny zbiór potencjalnie poprawnych danych wejściowych nosi nazwę słownika. Przy założeniu, że jesteśmy w stanie obliczać dwa miliony skrótów na sekundę, przegląd całego słownika zajmie około 30 godzin.

© copyright by Cyberskiller Sp. z o.o., 2020.
Wszystkie prawa zastrzeżone.
Wszelkie zmiany, kopiowanie i wykorzystywanie
dozwolone wyłącznie za uprzednią zgodą Cyberskiller Sp. z o.o.