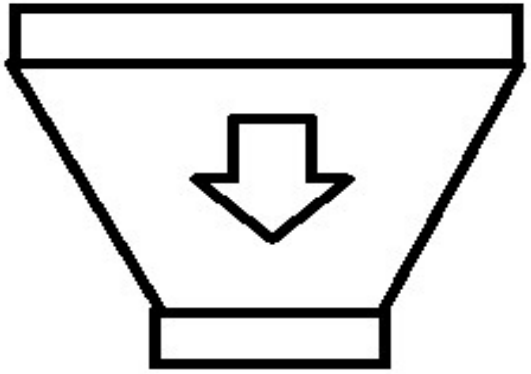
Funkcja skrótu

Czym jest?

Funkcja skrótu,

funkcja mieszająca,

funkcja haszująca



zastosowania

1. Detekcja błędów/Integralność pliku danych(przechowywanie/przesyłanie)
2. Identyfikacja pliku
3. Generowanie ciągów pseudolosowych/generowania unikalnych identyfikatorów?
4. Integralność aplikacji
5. Przechowywanie haseł
6. Utworzenie klucza
7. HMAC (hash-based message authentication code)
8. Proces wstępny dla podpisu elektronicznego
9. Blokchain

openssl help

OpenSSL

wieloplatformowa, otwarta implementacja

algorytmów kryptograficznych ogólnego przeznaczenia.

openssl help

Standard commands

asn1parse	ca	ciphers	cms
crl	crl2pkcs7	**dgst**	dhparam
dsa	dsaparam	ec	ecparam
enc	engine	errstr	gendsa
genpkey	genrsa	help	list
nseq	ocsp	passwd	pkcs12
pkcs7	pkcs8	pkey	pkeyparam
pkeyutl	prime	rand	rehash
req	rsa	rsautl	s_client
s_server	s_time	sess_id	smime
speed	spkac	srp	storeutl
ts	verify	version	x509

openssl dgst –help

- Usage: dgst [options] [file...]
- file... files to digest (default is stdin)
- -help Display this summary
- -list List digests
- -c Print the digest with separating colons
- -r Print the digest in coreutils format
- -out outfile Output to filename rather than stdout
- -passin val Input file pass phrase source
- -sign val Sign digest using private key
- -verify val Verify a signature using public key
- -prverify val Verify a signature using private key
- -signature infile File with signature to verify
- -keyform format Key file format (PEM or ENGINE)
- -hex Print as hex dump
- -binary Print in binary form
- -d Print debug info
- -debug Print debug info
- -fips-fingerprint Compute HMAC with the key used in OpenSSL-FIPS fingerprint
- -hmac val Create hashed MAC with key
- -mac val Create MAC (not necessarily HMAC)
- -sigopt val Signature parameter in n:v form
- -macopt val MAC algorithm parameters in n:v form or key
- -* Any supported digest
- -rand val Load the file(s) into the random number generator
- -writerand outfile Write random data to the specified file
- **-engine val** Use engine e, possibly a hardware device
- -engine_impl Also use engine given by -engine for digest operations

Ćwiczenie 1 - integralność

Znaleźć skrót z plik.txt o treści:

„Ala ma 100PLN.”

Ćwiczenie 1 - integralność

Znaleźć skrót z **plik.txt** o treści:

„Ala ma 100PLN.”

openssl dgst cw1.txt

echo -n "Ala ma 100PLN." | openssl dgst

Ćwiczenie 1a - integralność

Obliczyć skrót:

„Ala nie ma 100 PLN.”

```
echo -n "Ala nie ma 100PLN." | openssl dgst
```

Obliczyć skrót:

„ Ala ma 1000 PLN.”

```
echo -n "Ala ma 1000PLN." | openssl dgst
```

Ćwiczenie 1b

Zobacz jakie **algorytmy haszujące** ma openssl:

openssl dgst -list

Ćwiczenie 1c

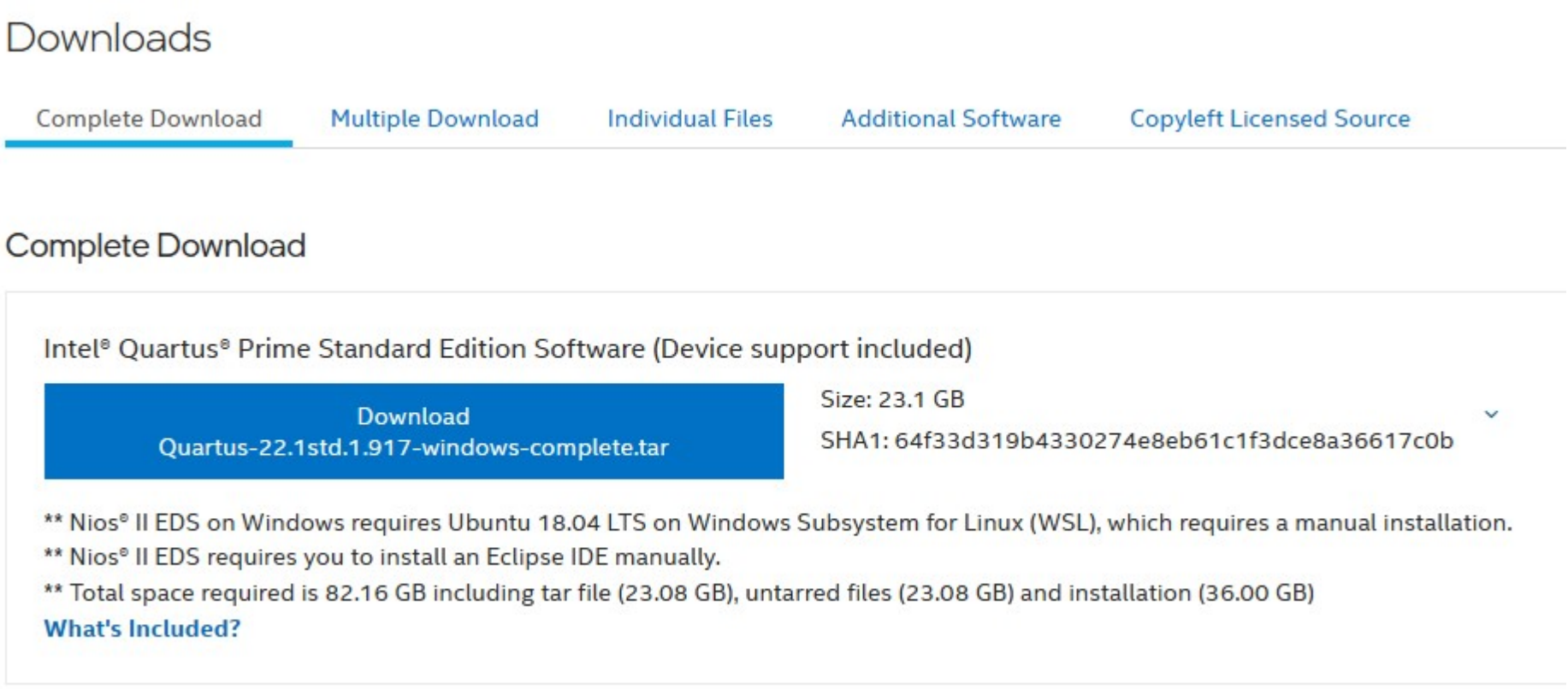
Znaleźć skrót z plik.txt o treści z wykorzystaniem MD5 i SHA512:

„Ala ma 100PLN.”

openssl dgst –md5 cw1.txt

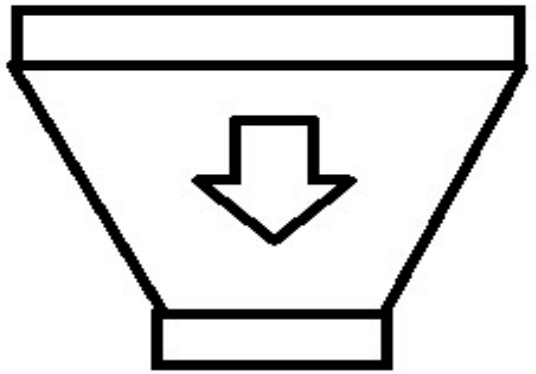
openssl dgst –sha512 cw1.txt

2 Identyfikacja pliku



Ćwiczenie 3 – ciąg pseudo

Wygeneruj ciąg pseudolowowy.



Ćwiczenie 3 – ciąg pseudo

Wygeneruj ciąg pseudolowowy.

```
echo -n "12345" | openssl dgst -sha256 –hex
echo -n "12346" | openssl dgst -sha256 –hex
echo -n "12347" | openssl dgst -sha256 -hex
```

4. Integralność aplikacji

- Integralność plików uruchomianych i wykorzystywanych przez apke przed uruchomieniem.

Ćwiczenie 5 – przechowywanie haseł

Znaleźć skróty dla haseł:

Stefan*2022

Stefan*2023

Ćwiczenie 5 – przechowywanie haseł

Znaleźć skróty dla haseł:

Stefan*2022

echo -n " Stefan*2022 " | openssl dgst

Stefan*2023

echo -n " Stefan*2023 " | openssl dgst

Ćwiczenie 5 - comment

Wyciek bazy **BEZ** funkcji skrótu:

Login;Hasło

Franek1; bialykruk1

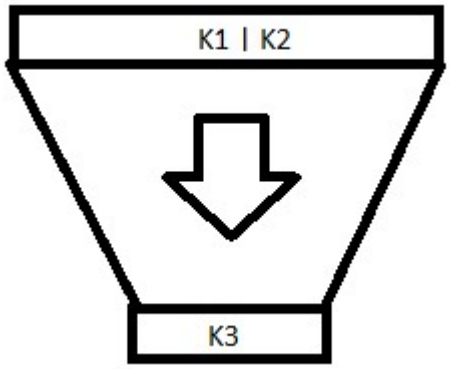
Wyciek bazy **Z** funkcją skrótu:

Login;Hasło

Franek1; 14ca89cc8724aed543f7671dfacb7dd706606ec75b2a23e2694cb1c291fdad38

Ćwiczenie 6 – utworzenie klucza

Utwórz wspólny klucz K3 na podstawie dwóch innych K1 i K2.



Ćwiczenie 7 - nienaruszalność

Prześlij plik tak, aby tylko odbiorca mógł zweryfikować czy nikt nie naruszył jego integralności.

Ćwiczenie 7

- openssl genpkey -algorithm RSA -out key1.pem
- openssl pkey -in key.pem -pubout -out pub_key1.pub
- openssl asn1parse -in pub_key1.pub
- openssl rand -hex 32 > klucz.hex
- openssl pkeyutl -encrypt -in klucz.hex -pubin -inkey pub_key1.pub -out ciph_key1.hex
- openssl pkeyutl -decrypt -in ciph_key.hex -inkey key1.pem -out deciph_key1.hex

- echo -n „Cześć W pierwszych słowach…" | openssl dgst -sha256 -mac hmac -macopt hexkey:01020304

Ćwiczenie 8 FS dla podpisu elektronicznego

Skrót jest zawsze:

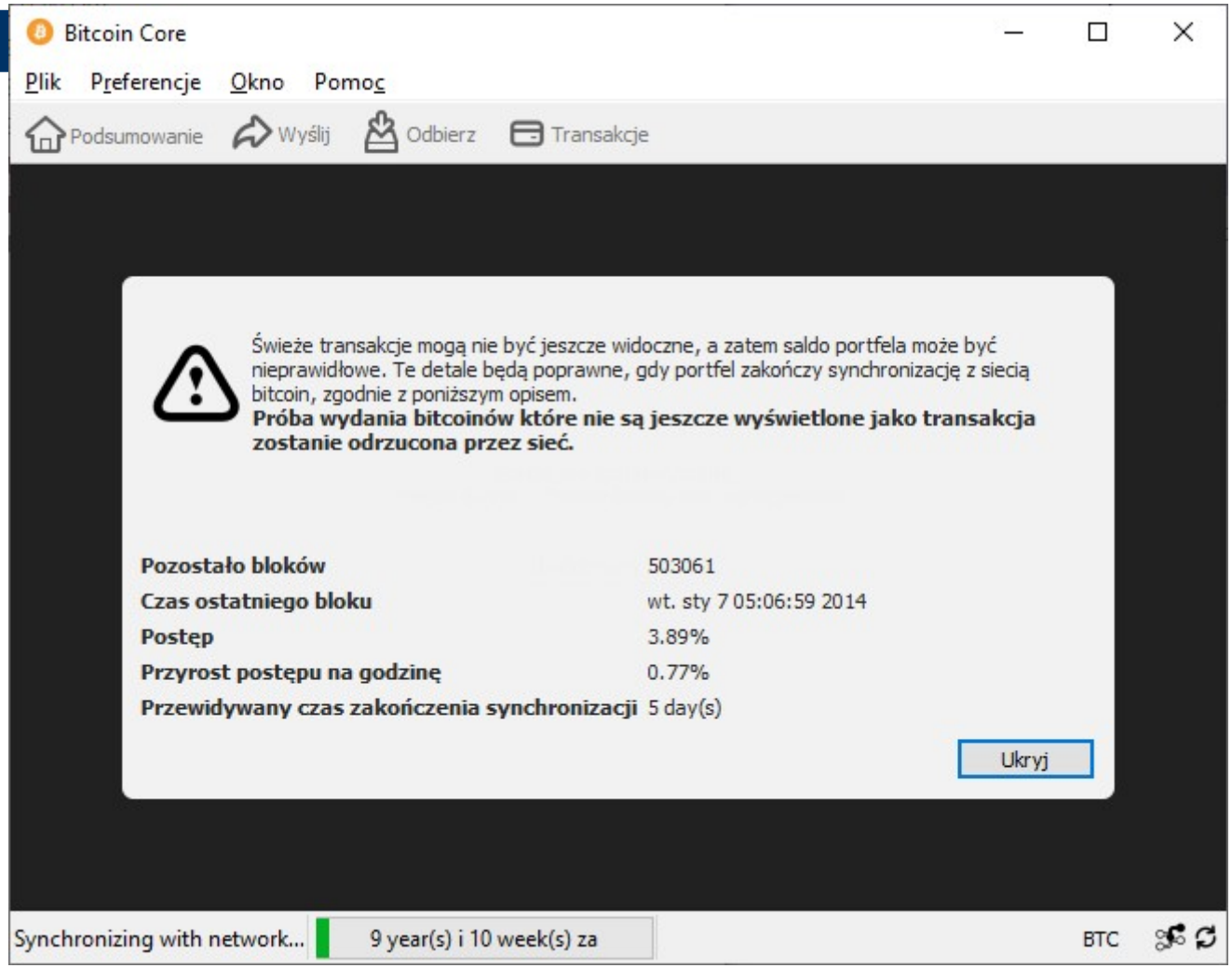
- o tej samej
- ograniczonej

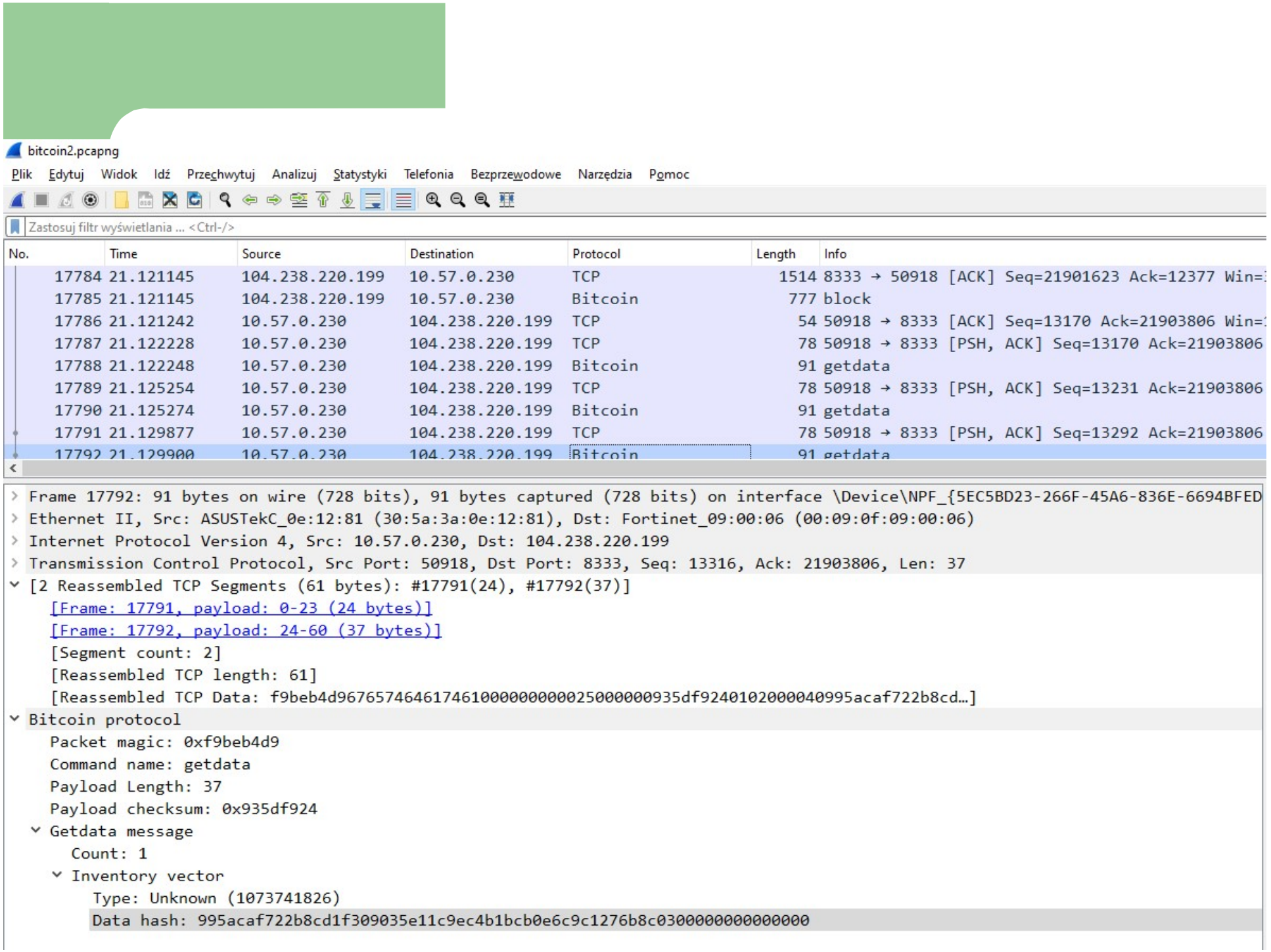
długości

9 FS w blockchain

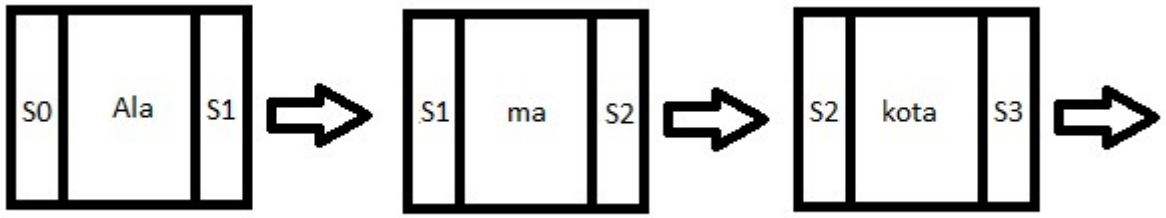
Blockchain (łańcuch bloków) – rosnąca lista rekordów, zwanych blokami, które powiązane są ze sobą przy użyciu kryptografii. Każdy blok składa się ze znaku czasowego, danych transakcji oraz kryptograficznego haszu (ang. *hash*) poprzedniego bloku, dzięki któremu formują one jednokierunkowy łańcuch, w którym tworzone bloki powiązane są ze wszystkimi wcześniejszymi[Wiki].

9 FS w blockchain




bitcoin2.pcapng
Plik Edytuj Widok Idź Przechwytuj Analizuj Statystyki Telefonia Bezprzewodowe Narzędzia Pomoc
Zastosuj filtr wyświetlania ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
17784 21.121145 104.238.220.199 10.57.0.230 TCP 1514 8333 → 50918 [ACK] Seq=21901623 Ack=12377 Win=
17785 21.121145 104.238.220.199 10.57.0.230 Bitcoin 777 block
17786 21.121242 10.57.0.230 104.238.220.199 TCP 54 50918 → 8333 [ACK] Seq=13170 Ack=21903806 Win=
17787 21.122228 10.57.0.230 104.238.220.199 TCP 78 50918 → 8333 [PSH, ACK] Seq=13170 Ack=21903806
17788 21.122248 10.57.0.230 104.238.220.199 Bitcoin 91 getdata
17789 21.125254 10.57.0.230 104.238.220.199 TCP 78 50918 → 8333 [PSH, ACK] Seq=13231 Ack=21903806
17790 21.125274 10.57.0.230 104.238.220.199 Bitcoin 91 getdata
17791 21.129877 10.57.0.230 104.238.220.199 TCP 78 50918 → 8333 [PSH, ACK] Seq=13292 Ack=21903806
17792 21.129900 10.57.0.230 104.238.220.199 Bitcoin 91 getdata
Frame 17792: 91 bytes on wire (728 bits), 91 bytes captured (728 bits) on interface \Device\NPF_{5EC5BD23-266F-45A6-836E-6694BFED
Ethernet II, Src: ASUSTekC_0e:12:81 (30:5a:3a:0e:12:81), Dst: Fortinet_09:00:06 (00:09:0f:09:00:06)
Internet Protocol Version 4, Src: 10.57.0.230, Dst: 104.238.220.199
Transmission Control Protocol, Src Port: 50918, Dst Port: 8333, Seq: 13316, Ack: 21903806, Len: 37
[2 Reassembled TCP Segments (61 bytes): #17791(24), #17792(37)]
[Frame: 17791, payload: 0-23 (24 bytes)]
[Frame: 17792, payload: 24-60 (37 bytes)]
[Segment count: 2]
[Reassembled TCP length: 61]
[Reassembled TCP Data: f9beb4d9676574646174610000000000250000000935df9240102000040995acaf722b8cd…]
Bitcoin protocol
Packet magic: 0xf9beb4d9
Command name: getdata
Payload Length: 37
Payload checksum: 0x935df924
Getdata message
Count: 1
Inventory vector
Type: Unknown (1073741826)
Data hash: 995acaf722b8cd1f309035e11c9ec4b1bcb0e6c9c1276b8c0300000000000000

9 FS w blockchain



KONIEC

??????????????????????????????????????
??????????????????????????????????????
??????????????????????????????????????
??????????????????????????????????????
??????????????????????????????????????
??????????????????????????????????????
??????????????????????????????????????
??????????????????????????????????????
??????????????????????????????????????