



Blockchain od podstaw

Wifi: GuestMansion

Hasło: GuestsAreWelcome

Link do spotkania: bit.ly/3GGE1Jt

Co będziemy dziś robić

- Kryptograficzne podstawy – funkcje hashu, kryptografia asymetryczna i podpisy cyfrowe
- Cyfrowa waluta od zera – jak zbudować cyfrową walutę od zera
- Ethereum i Starknet – jak zdecentralizować dowolny program i jak zrobić to wydajnie

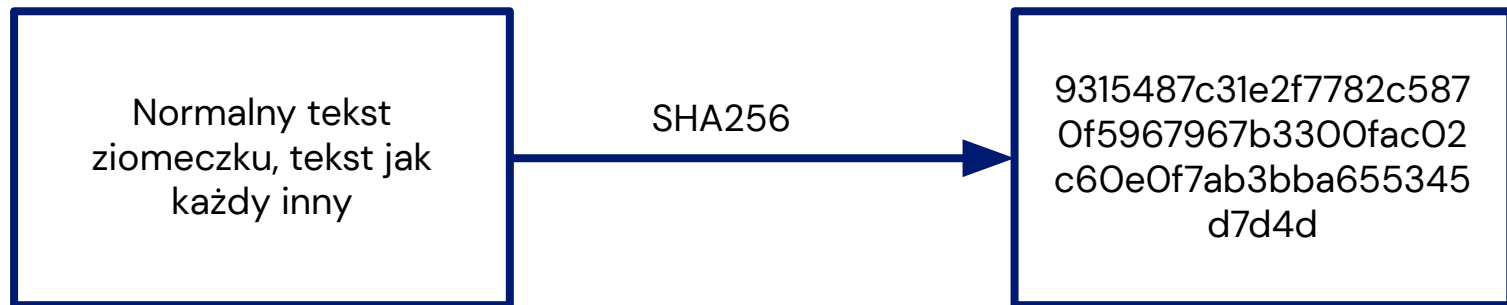


Kryptograficzne podstawy

Czego dowiesz się w tej części

- Kryptograficzne funkcje hashu – nieodwracalna kompresja informacji
- Szyfrowanie symetryczne i asymetryczne
- Podpisy cyfrowe

Kryptograficzne funkcje hashu



Właściwości kryptograficznej funkcji hashu

- Dowolna długość wejścia, stała długość wyjścia
- Jeżeli posiadamy hash tekstu to **nie da się** odtworzyć tekstu
- **Nie da się** znaleźć dwóch wejść które generują ten sam hash

Nie da się??



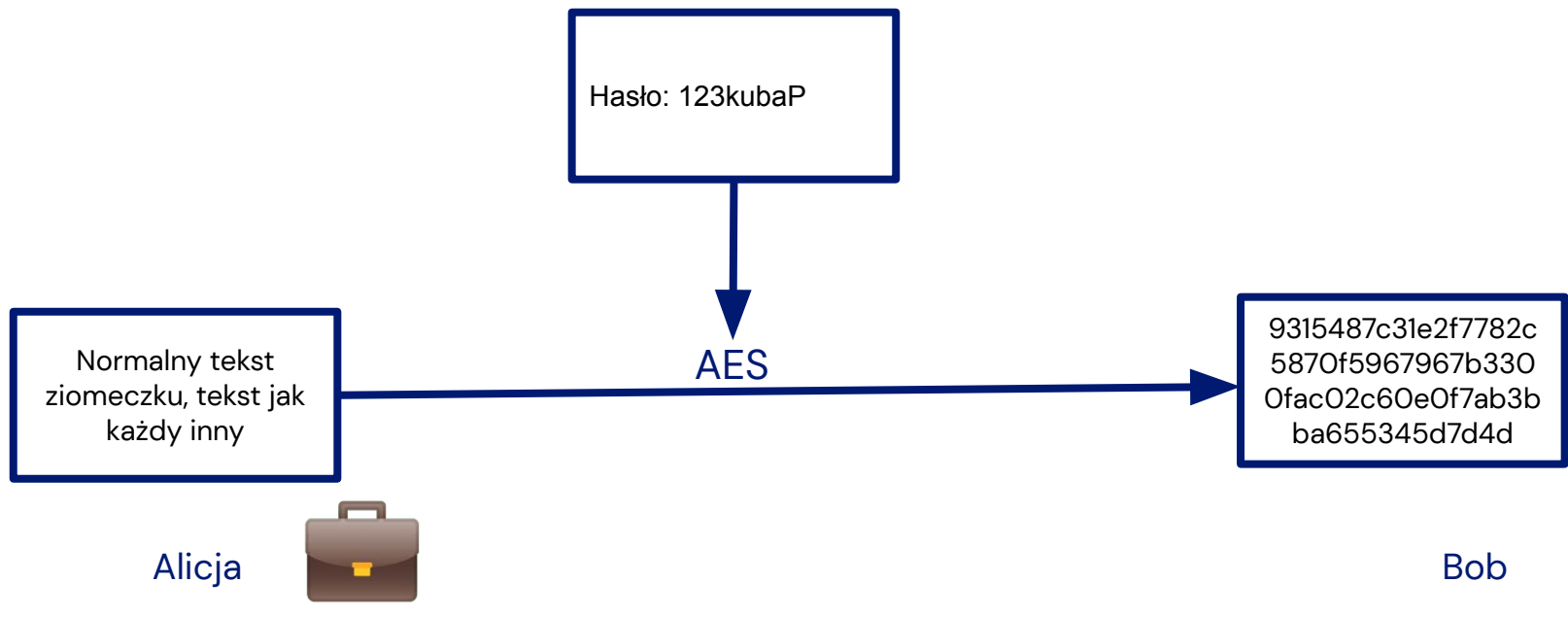
Przykład zastosowania

W bazie danych trzymamy zahashowane hasła użytkowników

Przy uwierzytelnianiu użytkownika hashujemy podane hasło i porównujemy hashe

W przypadku wycieku danych haseł w bazie nie da się ich odczytać

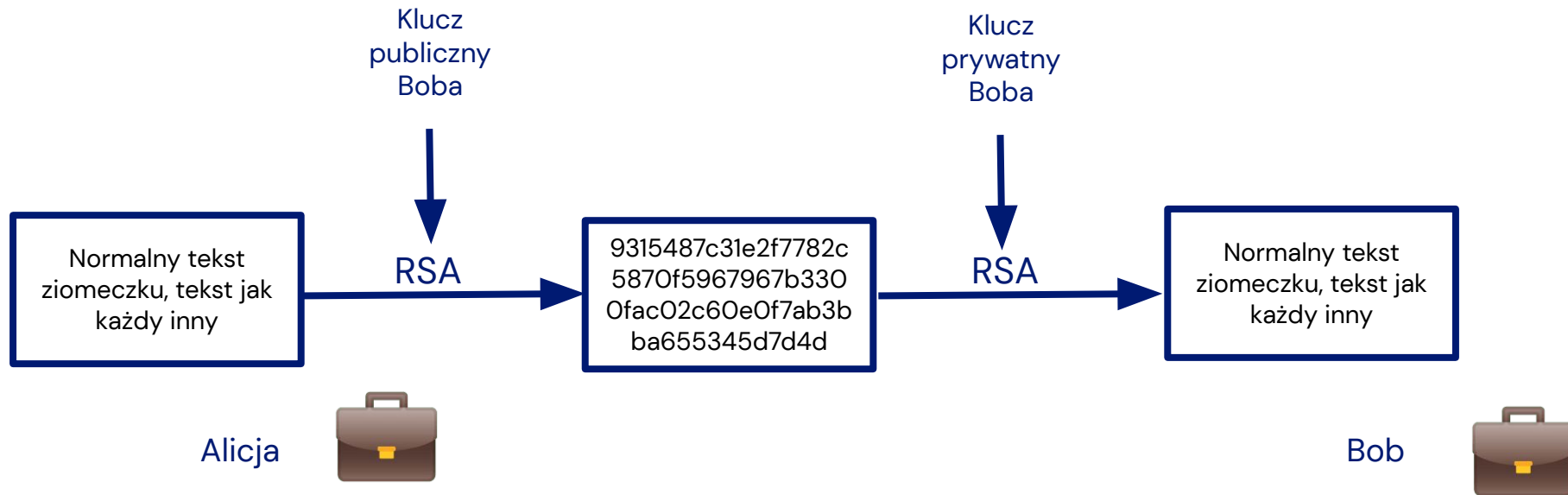
Kryptografia symetryczna



Problem?

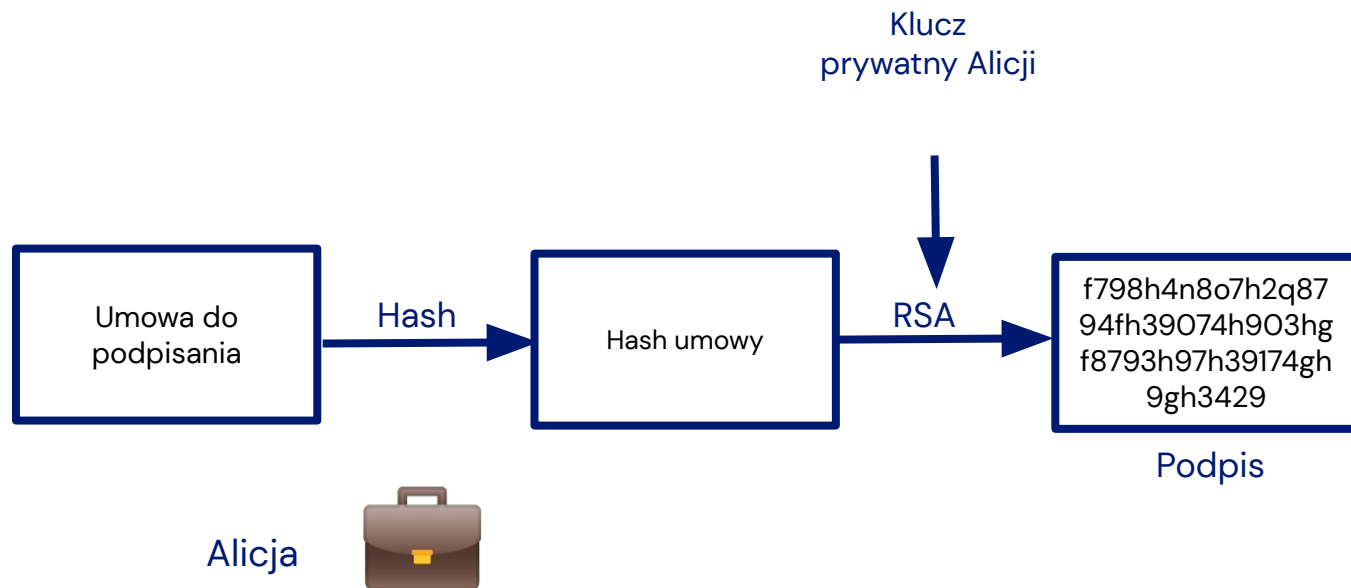
Obie strony muszą znać hasło

Kryptografia asymetryczna

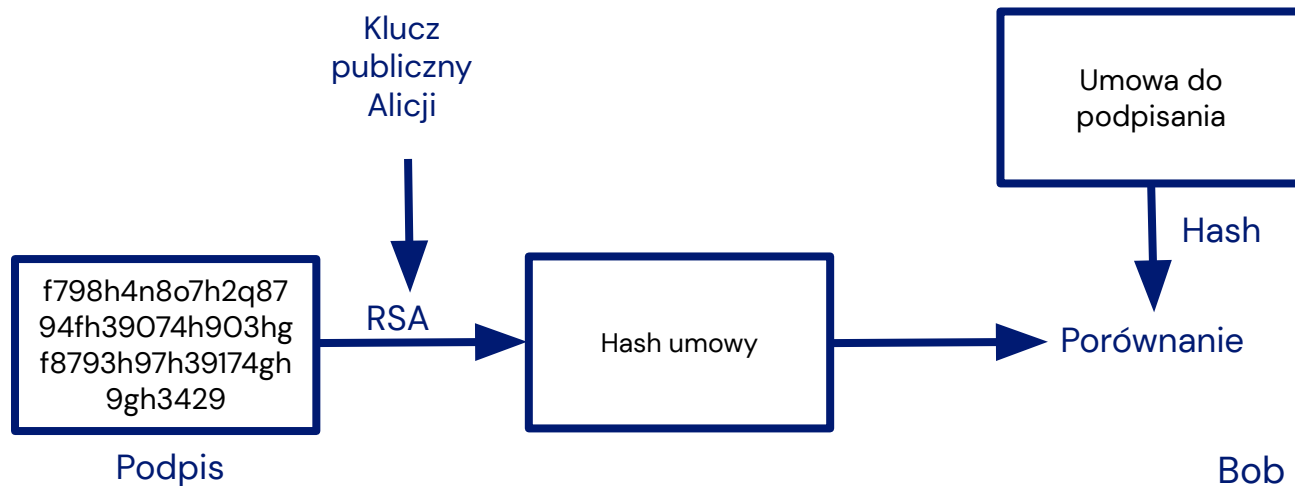


Nie da się cofnąć operacji szyfrowania
kluczem publicznym!

Podpis cyfrowy



Podpis cyfrowy



Ćwiczenie 1

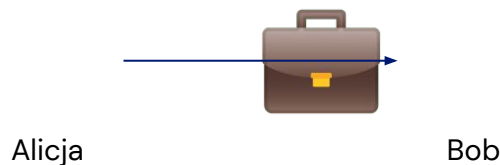
Proste schematy wykorzystujące podane metody kryptograficzne.

Link do repo: bit.ly/3FfptjE



Cyfrowa waluta

Najpierw... gotówka



Własności:

- Alicja nie może dać tego samego banknotu dwóm różnym osobom



Jeśli Alicja dała Bobowi banknot, to nie może tego cofnąć

- Wydawane przez bank centralny

Problem:

- Potwierdzenie autentyczności banknotu

SWMCoin



Chcemy stworzyć cyfrowego “coina” który:

- Jest przekazywalny
- Jest niepodrabialny
- W danym momencie ma tylko jednego właściciela

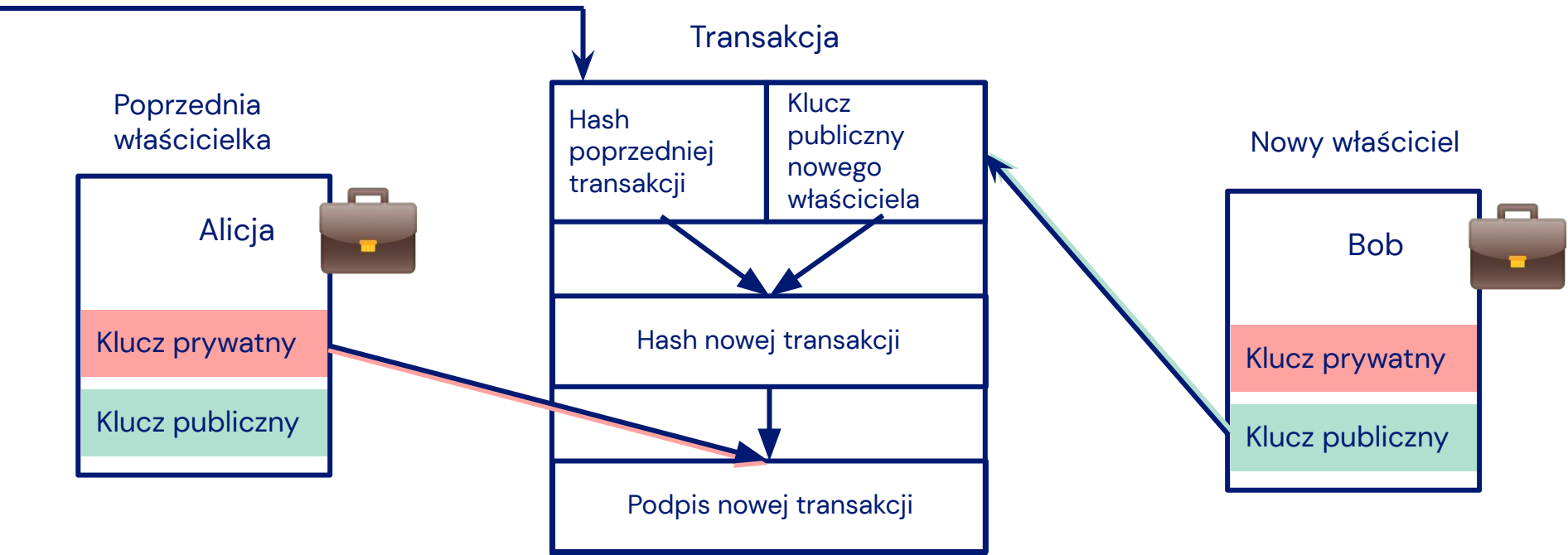
I co najważniejsze, działa w **zdecentralizowanym** systemie. Nie ma centralnej jednostki zarządzającej tą walutą.

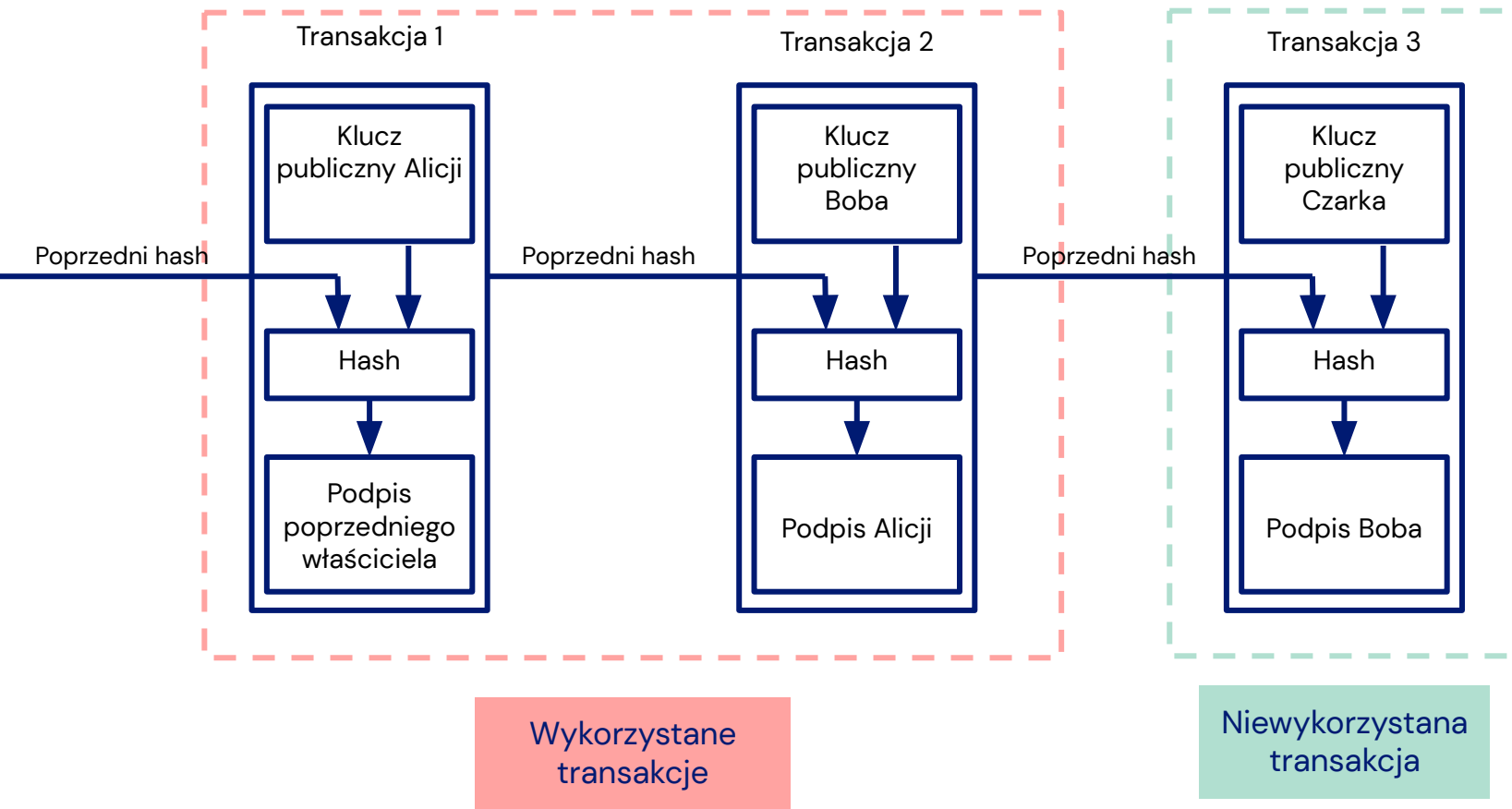
Dla uproszczenia zakładamy też, że nasz coin jest niepodzielny. W rzeczywistości Bitcoin dzieli się na 10^8 Satoshi.

Jak wejść w posiadanie SWMCoina?

Trzeba otrzymać go w transakcji.







Double spending

- Podwójne wydanie tego samego coina
- Nie istnieje w transakcjach gotówkowych – jedną monetę można przekazać komuś tylko raz
- W transakcjach internetowych rozwiązywany jest przez bank, który pilnuje, żeby pieniądze były wydawane tylko raz
- W świecie kryptowalut, gdzie zaufana jednostka nie istnieje, podwójne wydawanie tego samego coina jest możliwe

Publiczny rejestr transakcji

Tworzymy publiczny rejestr, w którym publikowane są wszystkie transakcje.

Każdy może dodać transakcję,
podpisy transakcji są weryfikowane.
Mamy zachowaną kolejność
transakcji.

Problem? 1 rejestr = pełna
centralizacja

0.  Alicja przekazuje 1 SWMCoin do  Boba
1.  Alicja przekazuje 1 SWMCoin do  Czarka
2.  Czarek przekazuje 1 SWMCoin do  Boba
3.  Bob przekazuje 1 SWMCoin do  Alicja
4.  Bob przekazuje 1 SWMCoin do  Doroty

•
•
•

Skąd wziąć pierwsze transakcje w systemie?

Rozwiązanie tymczasowe – pierwsze n transakcji tworzy “bank”.

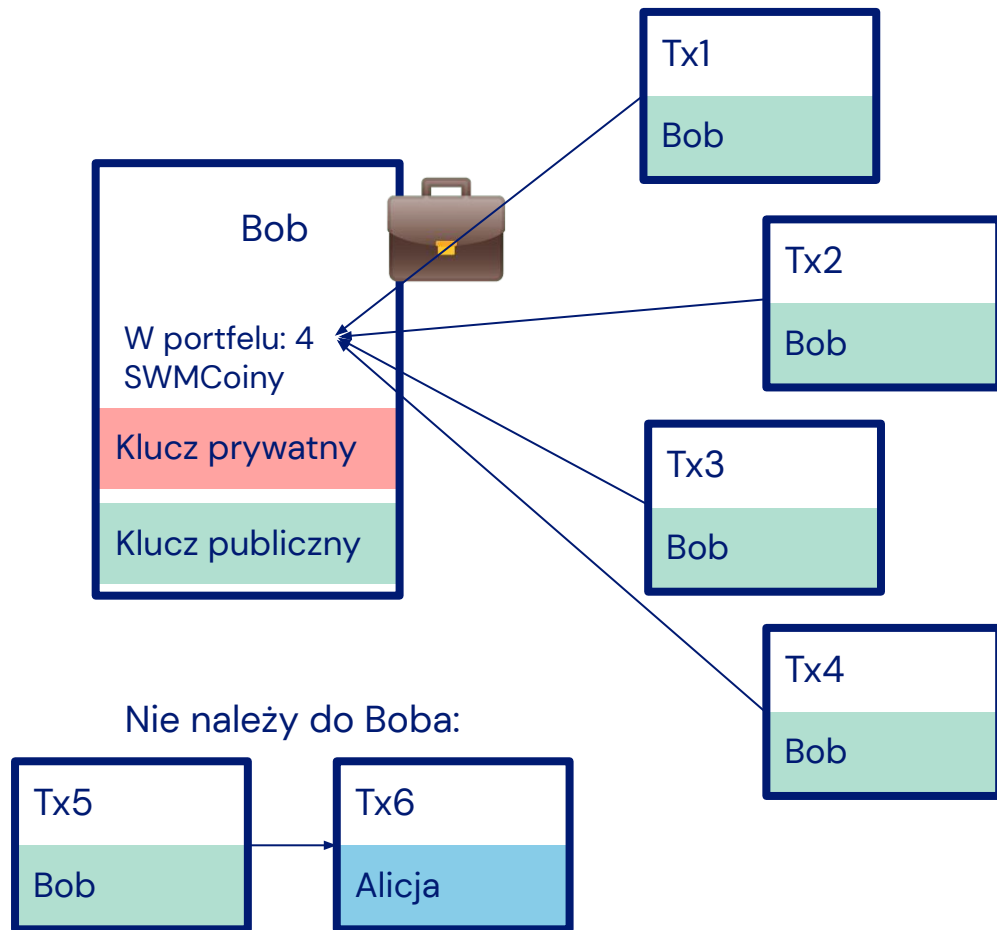
0.  Bank przekazuje 1 SWMCoin do  Alicji
1.  Bank przekazuje 1 SWMCoin do  Alicji
2.  Bank przekazuje 1 SWMCoin do  Czarka
3.  Alicja przekazuje 1 SWMCoin do  Boba
4.  Alicja przekazuje 1 SWMCoin do  Czarka
5.  Czarek przekazuje 1 SWMCoin do  Boba
6.  Bob przekazuje 1 SWMCoin do  Alicji
7.  Bob przekazuje 1 SWMCoin do  Doroty

Jak działa portfel?

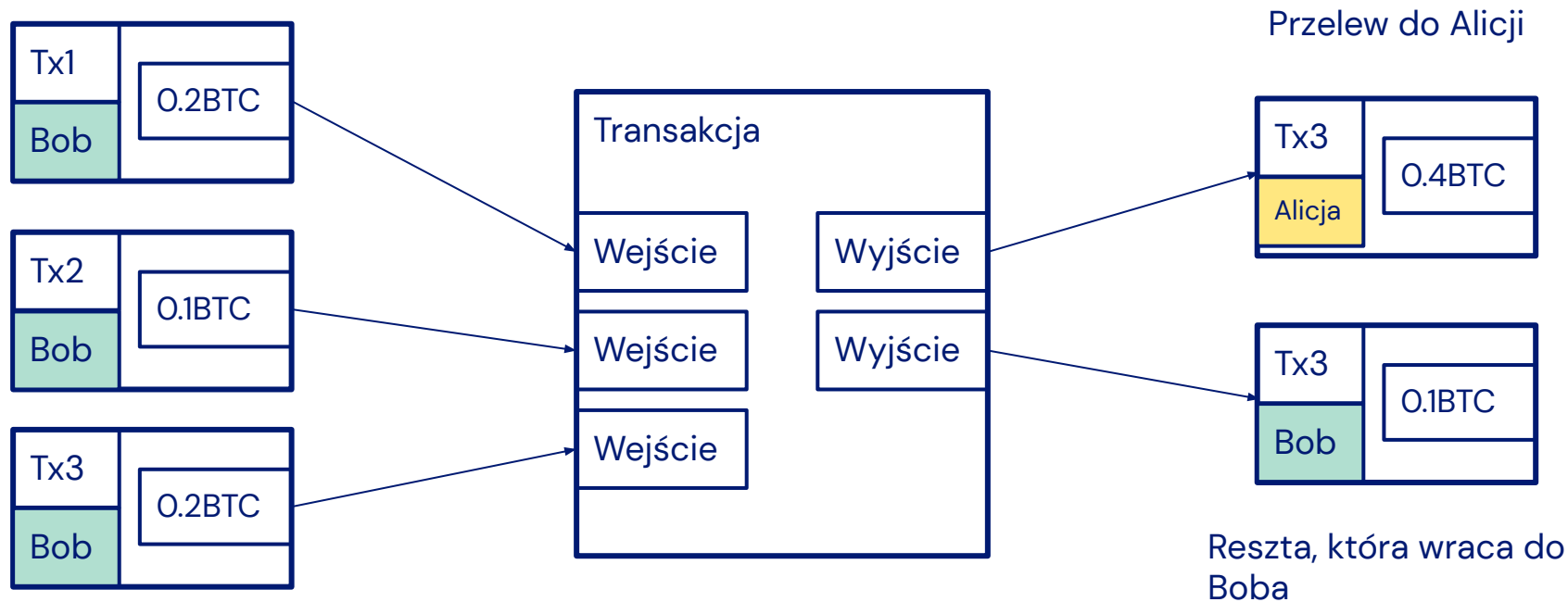
Portfel przechowuje prywatny i publiczny klucz użytkownika.

Zlicza wszystkie niewykorzystane transakcje wysłane do Boba.

Pozwala także na stworzenie i podpisanie transakcji, a następnie wysłanie jej do rejestru.



Bonus: Podział coina, jak to robi Bitcoin?



Ćwiczenie 2

Implementujemy metody w dwóch klasach *TransactionRegistry* i *Wallet*. Ich funkcjonalność jest następująca:

TransactionRegistry:

przechowuje transakcje, przyjmuje nowe transakcje i weryfikuje czy są poprawne.

Wallet:


Przechowuje prywatny i publiczny klucz użytkownika. Sprawadza stan konta użytkownika i pozwala na podpisywanie transakcji i wysyłanie ich do rejestru transakcji.

Decentralizacja – dlaczego jest potrzebna?


- właściciel rejestru może dowolnie dodawać i usuwać transakcje
 - lub tworzyć nowe coiny! 😬
-
- w przypadku awarii serwera wszystkie dane są tracone
 - właściciel nie ma korzyści z utrzymywania rejestru

Jak rozwiązać ten problem?

Zdecentralizowany rejestr transakcji

- pozwala stwierdzić kolejność transakcji  **Było!**
- określa czas w jakim dana transakcja została wykonana
- każdy użytkownik sieci przechowuje swoją kopię rejestru
- wszyscy użytkownicy muszą zgadzać się co do stanu rejestru – konsensus

Zdecentralizowany rejestr transakcji

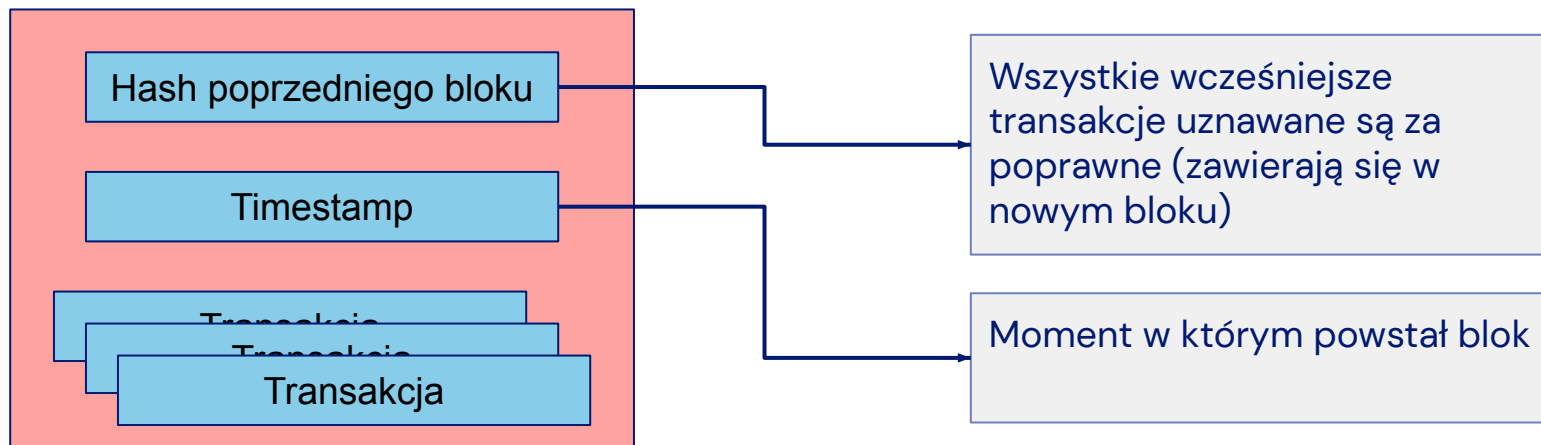
- pozwala stwierdzić kolejność transakcji  **Było!**
- określa czas w jakim dana transakcja została wykonana
- każdy użytkownik sieci przechowuje swoją kopię rejestru
- wszyscy użytkownicy muszą zgadzać się co do stanu rejestru – konsensus

Jak wygląda struktura danych takiego rejestru?

Element I – blok

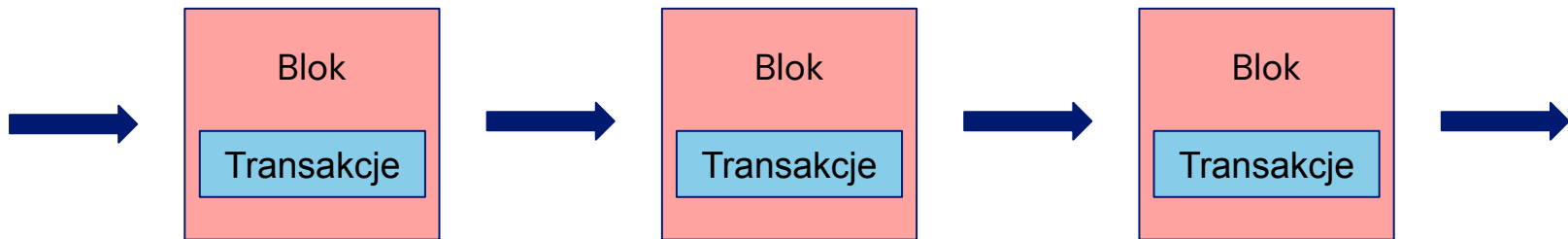
- hash poprzedniego bloku – wszystkie wcześniejsze transakcje są uznawane za poprawne w nowym bloku
- transakcja – transakcja, która miała miejsce od momentu stworzenia poprzedniego bloku
- timestamp – czas w którym blok został stworzony
- nonce – o tym za chwilę

Element I – blok



Element II – łańcuch bloków!

- bloki tworzone są w jakiejś kolejności
- można ustawić je w uporządkowany sposób
- rejestr jest budowany blok po bloku



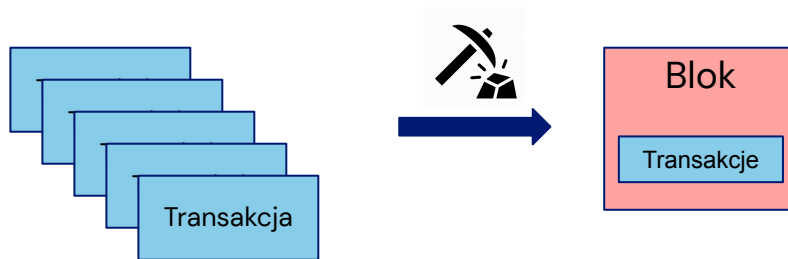
Kłopoty!

- Kto ma tworzyć nowe bloki?
- Skąd wiadomo, że ktoś nie oszukuje?



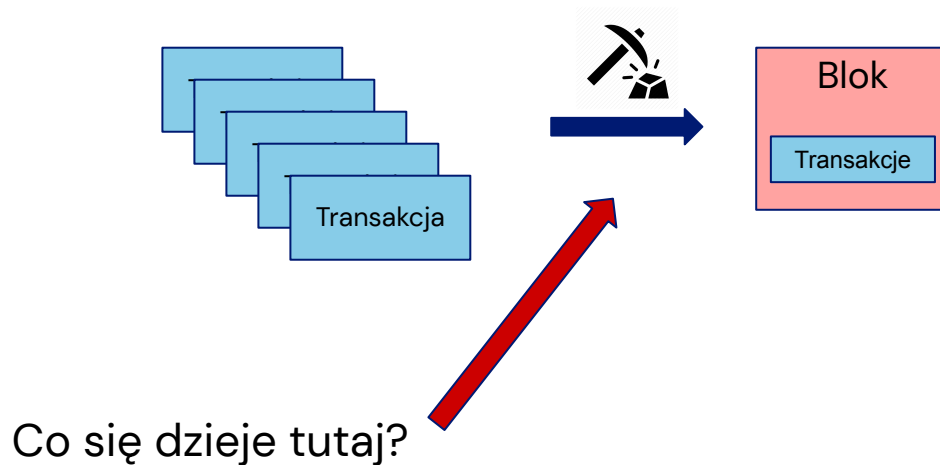
Kopanie 🎉

- kopanie – proces tworzenia nowego bloku



Kopanie 🎉

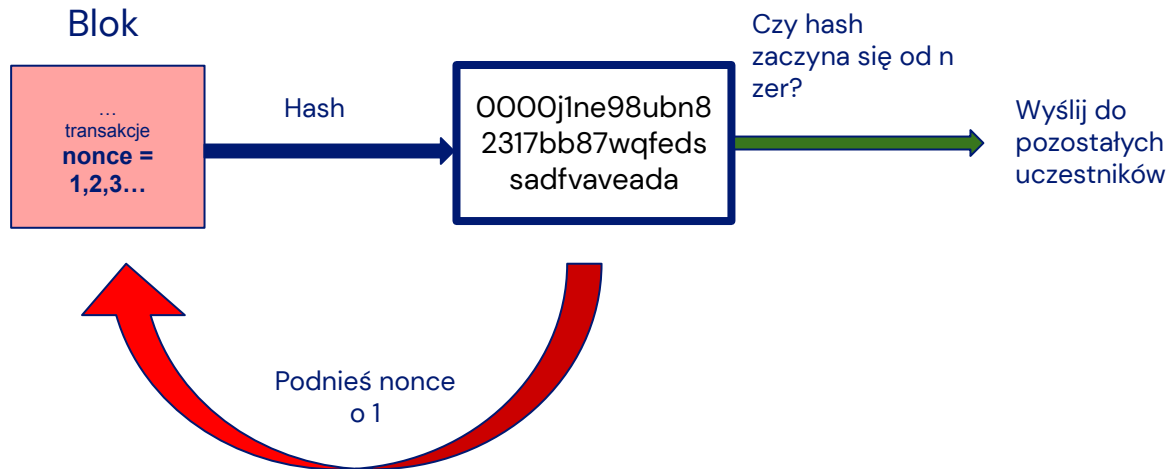
- kopanie – proces tworzenia nowego bloku



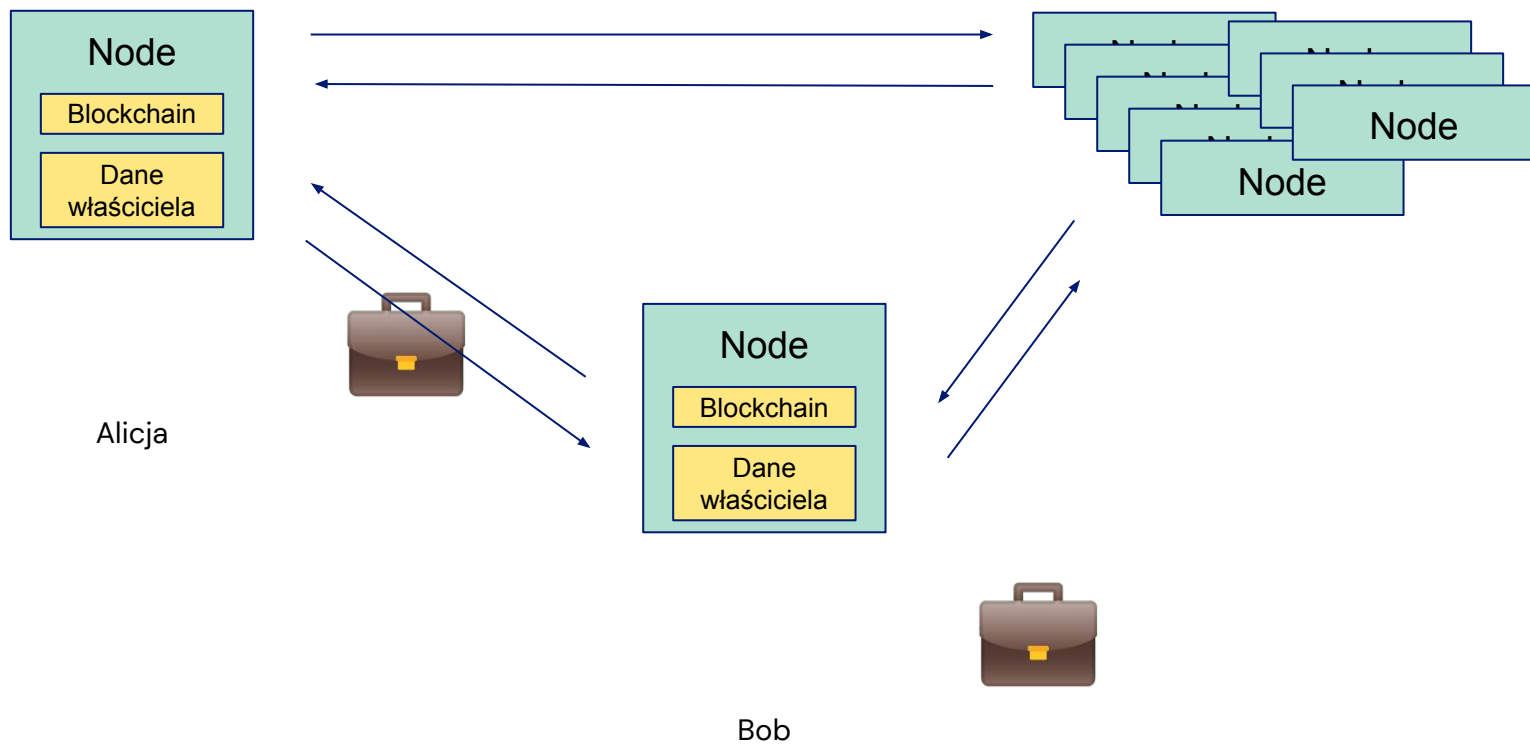
Kopanie krok po kroku

- górnik zbiera transakcje
- wykorzystując transakcje i informacje z wcześniejszego bloku, tworzy nowy blok
- hash każdego z bloków musi spełniać określone warunki (np. musi być mniejszy niż jakaś z góry określona wartość) – powiedzieć dlaczego
- aby można było manipulować hashem potrzebujemy jakiejś zmiennej wartości, którą w przypadku bloku jest **nonce**
- górnik sprawdza kolejne wartości nonce, dopóki nie znajdzie takiej dla której hash spełnia określone wymagania

Kopanie krok po kroku – diagram

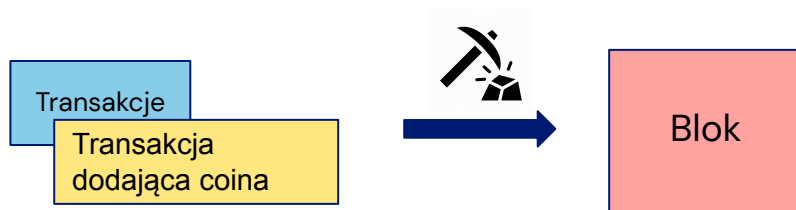


Ogólny diagram sieci



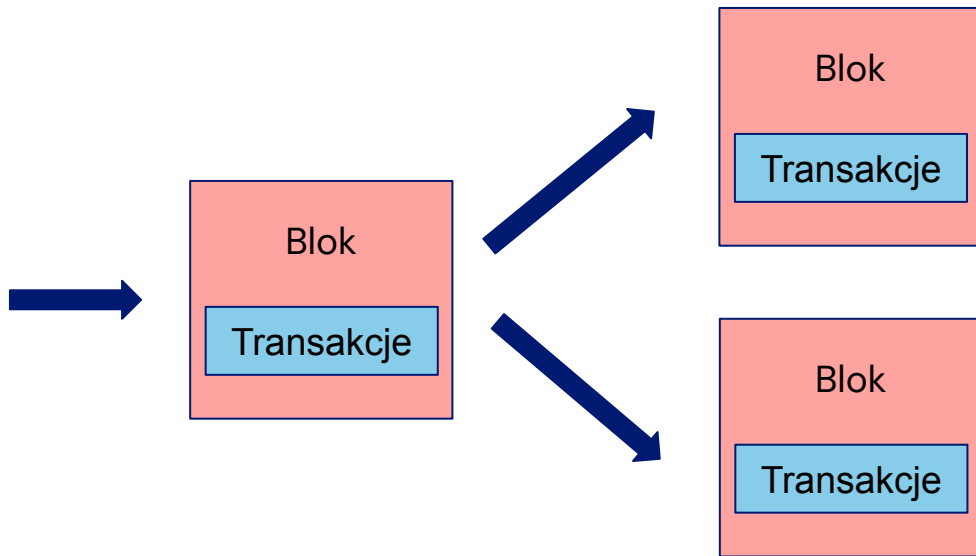
Co ja z tego mam?

- za każdy wykopany blok górnik otrzymuje nowego coina
- jak?
- poza transakcjami zebranymi z sieci, tworzy transakcję, która tworzy nowego coina i dodaje go do swojego konta
- inne, uczciwe, węzły się na to zgadzają



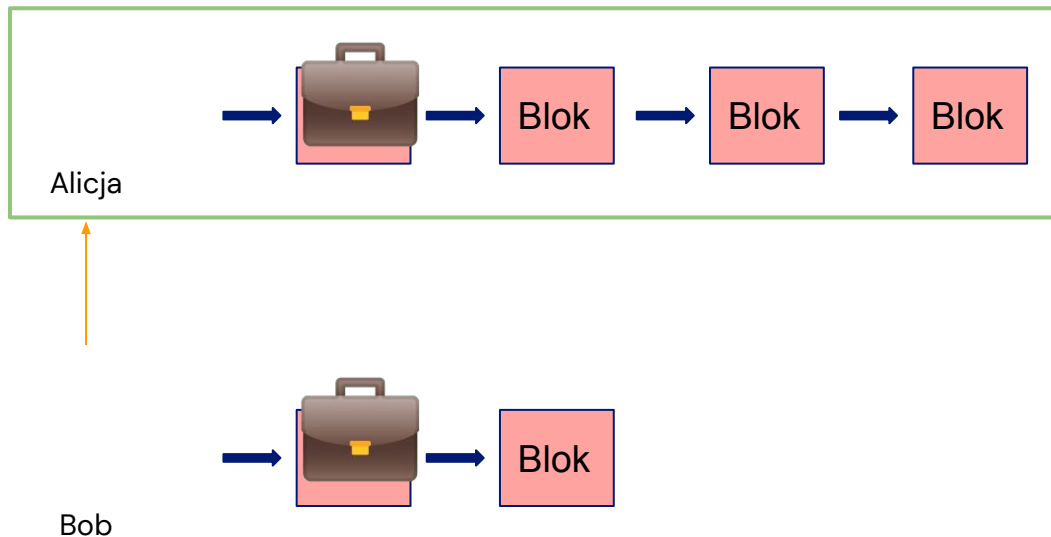
Rozgałęzienia

- Co jeśli dwóch górników wykopie blok w tym samym czasie?



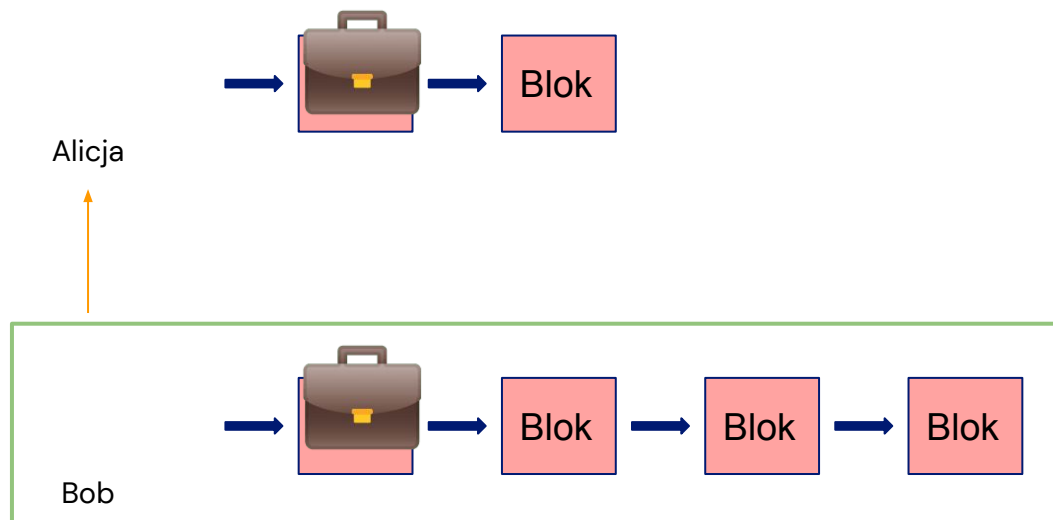
Konsensus – przypadek I

- algorytm podejmowania decyzji, który z łańcuchów jest poprawny
- kryterium to długość łańcucha



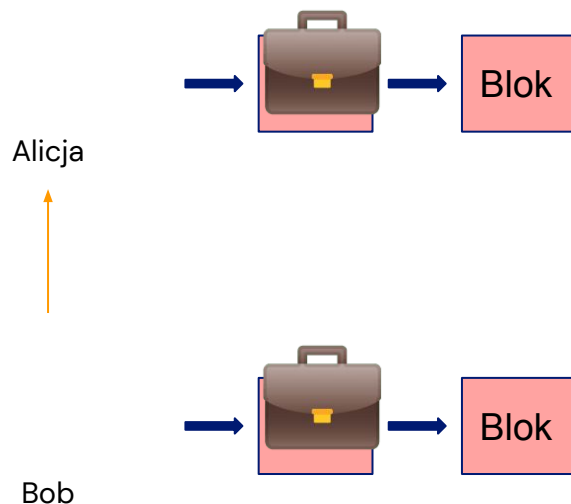
Konsensus – przypadek II

- algorytm podejmowania decyzji, który z łańcuchów jest poprawny
- kryterium to długość łańcucha

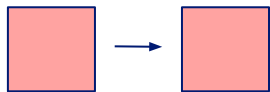


Konsensus – przypadek III

- algorytm podejmowania decyzji, który z łańcuchów jest poprawny
- kryterium to długość łańcucha



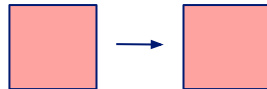
Dlaczego to wszystko ma sens?



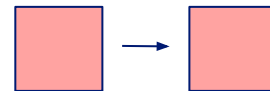
Alicja



1. Alicja płaci
Bobowi



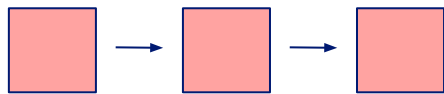
Inne węzły



Bob



Dlaczego to wszystko ma sens?



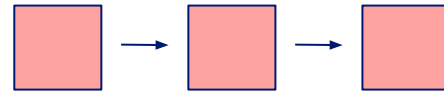
Alicja



2. Transakcja jest
dodawana i Bob
wysyła produkt



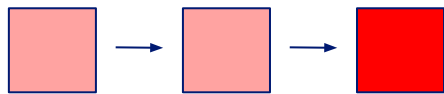
Inne węzły



Bob



Dlaczego to wszystko ma sens?

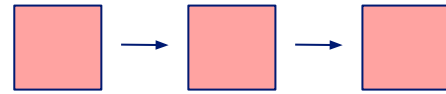


Alicja

3. Alicja wyrzuca
blok z transakcją
i tworzy nowy,
nieprawdziwy

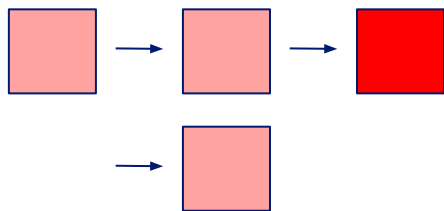


Inne węzły



Bob

Dlaczego to wszystko ma sens?



Alicja



4. Aby przekonać sieć, że jej wersja jest poprawna Alicja musi kopać szybciej niż wszyscy razem wzięci



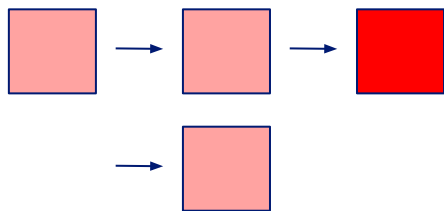
Inne węzły



Bob

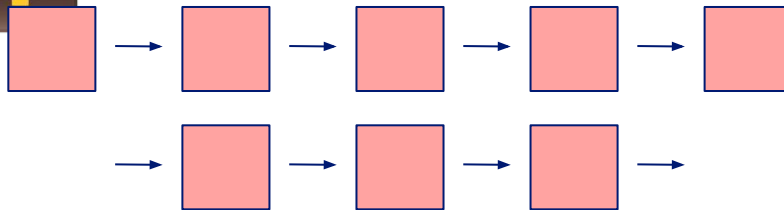


Dlaczego to wszystko ma sens?

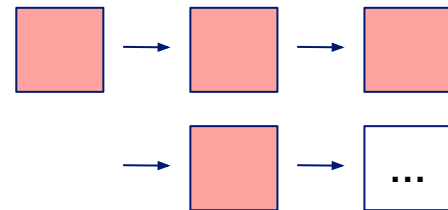


Alicja

5. Nie jest to możliwe,
chyba, że posiada ona 51%
mocy obliczeniowej sieci



Inne węzły

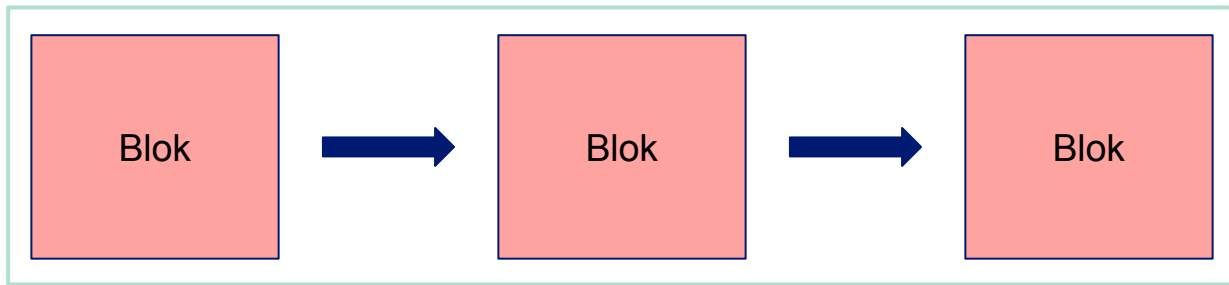


Bob



Elementy składowe blockchainu – podsumowanie

Struktura danych



Kopanie 🪓

Konsensus 🤝

Ćwiczenie 3

Tworzymy prosty model łańcucha bloków oraz uproszczoną implementację węzła, które pokrywają podstawowe koncepcje.



Ethereum

Po co nam kolejny blockchain i jak tworzyć zdecentralizowane aplikacje

Ethereum – tutaj zbudujesz co tylko chcesz

- Wykonywanie dowolnego kodu w sieci blockchain
- Solidity – wbudowany Turing-complete język programowania
- Ethereum można traktować jak maszynę na której odpalasz swój kod – nie trzeba samemu stawiać infrastruktury

**Jak wykorzystać możliwości
oferowane przez ETH?**

Najbardziej popularne typy kontraktów/aplikacji

- aplikacje finansowe
 - własne tokeny
 - portfele oszczędnościowe
- aplikacje półfinansowe
 - wypłata wynagrodzenia za usługę
- niezwiązane z pieniędzmi
 - głosowanie online
 - zdecentralizowana kontrola

Interaktywna platforma do nauki Solidity

Kurs CryptoZombies – <https://cryptozombies.io>

- Solidity od podstaw
- Edytor kodu w przeglądarce – nie trzeba setupować środowiska
- Rozwijanie aplikacji w kolejnych lekcjach
- Dokładne opisy zadań i wyjaśnianie koncepcji
- Ciężko utknąć (w razie czego są też odpowiedzi)

- Za darmo!



Dlaczego ETH się nie skaluje?

Wysokie koszty transakcji – skąd się biorą

Każdy node przetwarza wszystkie dane

- każdy musi mieć kopię całego łańcucha
- do każdego węzła przychodzi każda transakcja
- zwiększenie liczby węzłów nic nie daje, bo nie można podzielić pomiędzy nie pracy

Większa aktywność – większe koszty

Ustawia się fee dla transakcji (ilość ETH, które chcemy wydać na pokrycie kosztów transakcji).

Górnicy wybierają transakcje z największym fee, stąd im większa aktywność tym większe koszty.

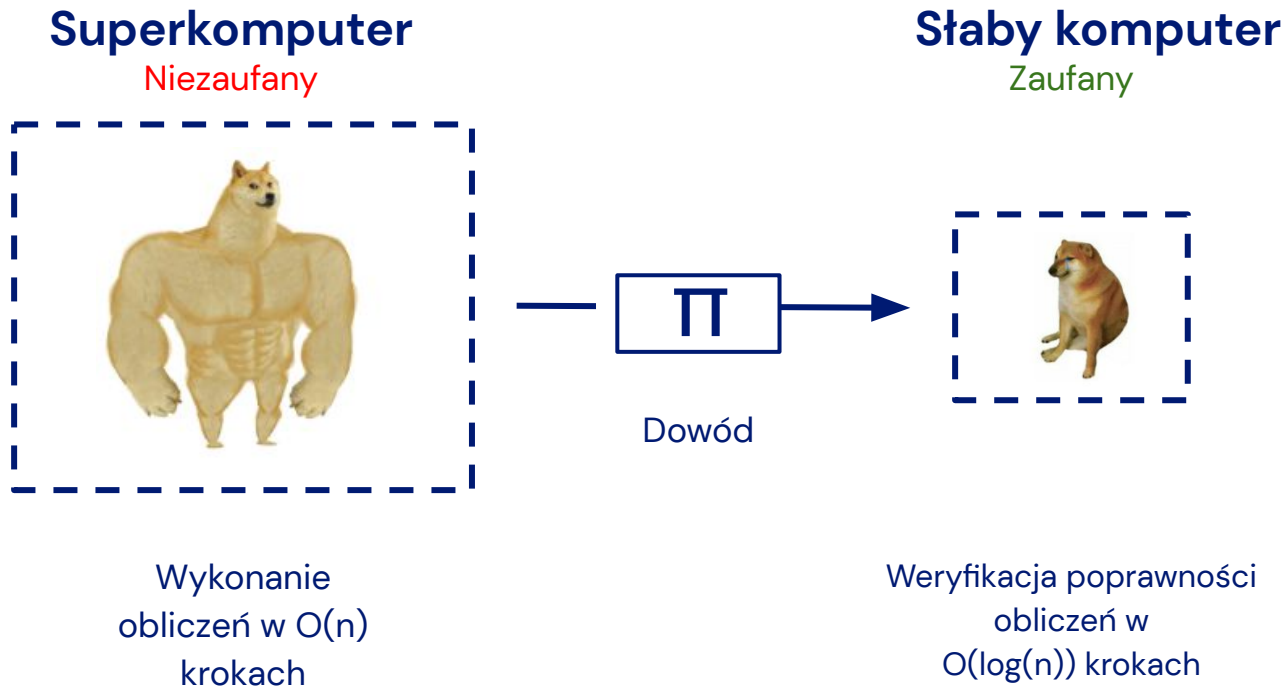
Jeśli chcesz, żeby transakcja wykonana się szybko to musisz zwiększyć fee.



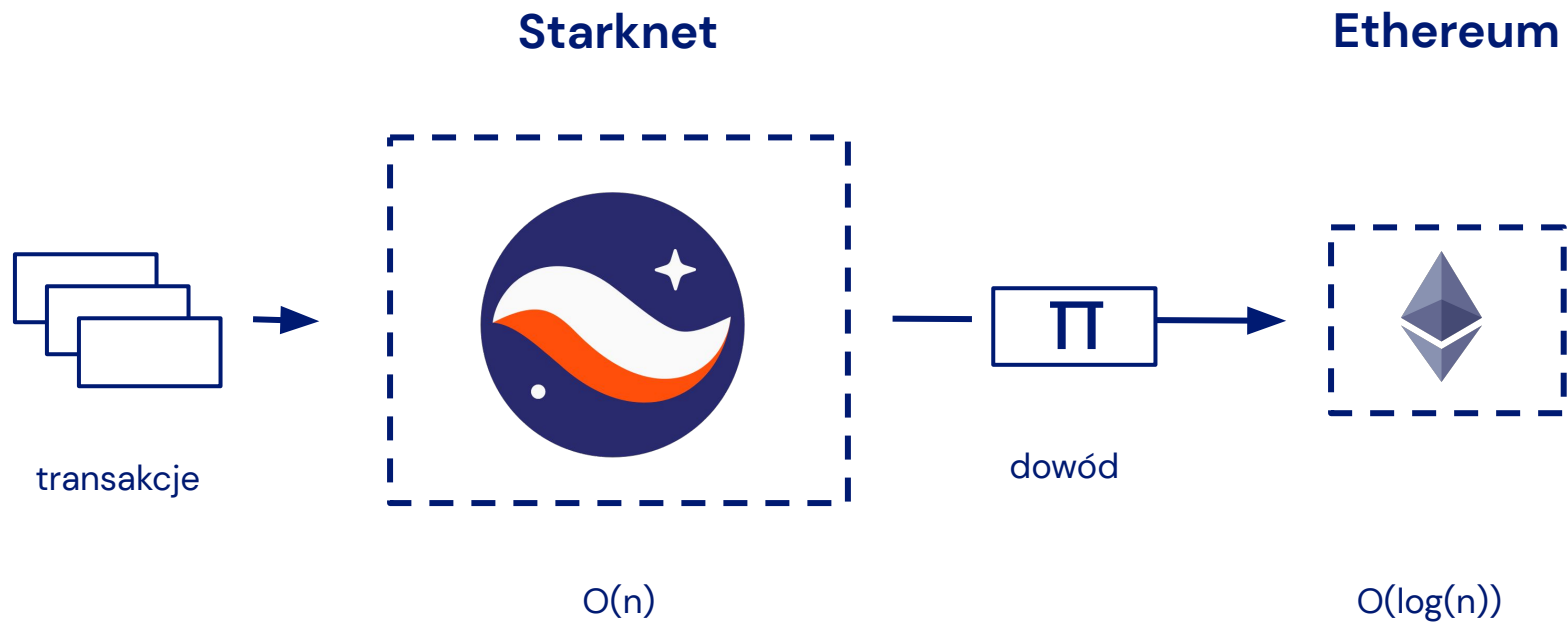
STARKNET

Warstwa druga Ethereum
Czyli co robimy w Software Mansion

Czym są dowody STARK



Starknet



Język Cairo

Język do smart kontraktów na Starknecie.

Pozwala na efektywne udowadnianie poprawności obliczeń.



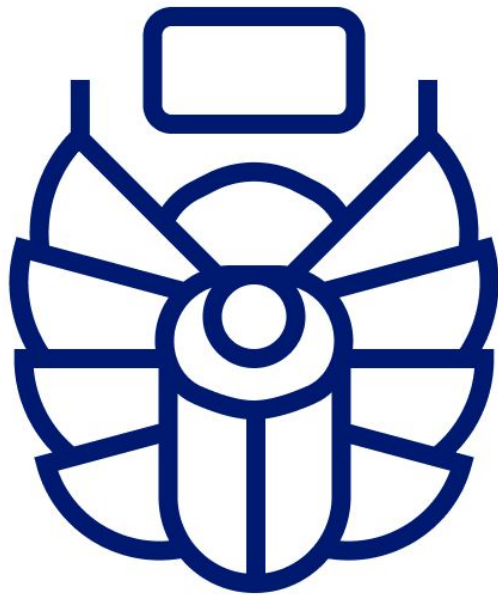
Co robimy w Software Mansion?



Protostar

- test runner do smart kontraktów napisanych w Cairo
- lokalne instancja Starknet zoptymalizowana pod wykonywanie testów
- rozszerzenia do kompilatora Cairo
- Python i Rust

Co robimy w Software Mansion?



Scarb

- package manager do Cairo
- centrum ekosystemu
- oparty na wtyczkach
- Rust

Co robimy w Software Mansion?

- Interakcja ze Starknetem z poziomu różnych języków



Swift



SDK Python, JVM, Swift

Staze letnie!

- Koniec marca/początek kwietnia
- React Native, Blockchain, Elixir i inne
- Praca nad rzeczywistymi projektami

<http://bitly.pl/P4RBk>



Newsletter