ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY W SZCZECINIE

WYDZIAŁ INFORMATYKI

mgr inż. Kamil Kaczyński

Zastosowanie zaawansowanych systemów blockchain do wzmacniania odporności systemów informatycznych

Rozprawa doktorska

Promotor: prof. dr hab. n. mat. inż. Jerzy Gawinecki

Szczecin 2024

Streszczenie

W rozprawie określono możliwości wykorzystania zaawansowanych technologii blockchain do wzmacniania odporności systemów informatycznych. Autor dokonał przeglądu istniejących technologii, ze szczególnym uwzględnieniem możliwości ich wykorzystania do zbudowania usługi zdecentralizowanego przechowywania danych z uwierzytelnionym dostępem. W pracy przedstawiono schematy wykorzystania technologii rejestru rozproszonego w systemach informatycznych, wskazując na możliwości poprawy parametrów integralności i dostępności danych, a także realizacji scenariuszy obliczeń rozproszonych. Autor przedstawił schemat uwierzytelnionego dostępu do danych w środowisku rozproszonym, wraz ze wskazaniem możliwości jego praktycznej implementacji. Ponadto opisane zostały wyniki ewaluacji zaproponowanego schematu, wskazujące na gwarantowane przez nie usługi kryptograficzne.

Słowa kluczowe: blockchain, decentralizacja, IPFS, DLT, kryptografia

Abstract

In the dissertation, the possibilities of using advanced blockchain technologies to enhance the resilience of information systems are presented. The author reviewed existing technologies, with particular emphasis on their potential for building a decentralized data storage service with authenticated access. The paper proposed schemes for using distributed ledger technology in information systems, pointing to the possibilities of improving data integrity and availability parameters, as well as implementing distributed computing scenarios. The author presented a scheme for authenticated data access in a distributed environment, along with the possibility of its practical implementation. In addition, the results of the evaluation of the proposed scheme were presented, indicating the cryptographic services guaranteed by it.
Keywords: blockchain, decentralization, IPFS, DLT, cryptography

Spis treści

Wykaz skrótów

AES – Advanced Encryption Standard

AMM – Automated Market Maker

BFN – Bysantine Fault Node

BFT – Bysantine Fault Tolerance

BI – Block Index

BTC – Bitcoin

CDN – Content Deliver Network

CFN – Crash Fault Node

CFT – Crash Fault Tolerance

CID – Content Identifier

DAG – Directed Acyclic Graph

DAO – Decentralized Autonomous Organization

DBFT – Distributed Bysantine Fault Tolerance

DDoS – Distributed Denial of Service

DeFi – Decentralized Finance

DEX – Decentralized Exchange

DHT – Distributed Hash Table

DLT – Distributed Ledger Technology

DNS – Domain Name System

DoS – Denial of Service

DPoS – Delegated Proof of Stake

DSHT – Distributed Sloppy Hash Table

ECIES – Elliptic Curve Integrated Encryption Scheme

EIP – Etehereum Improvement Proposal

ERC – Ethereum Request for Comments

ETH – Ethereum

EVM – Ethereum Virtual Machine

GCM – Galois Counter Mode

GPS – Global Positioning System

HKDF – HMAC-based Key Derivation Function

HLF – HyperLedger Fabric

HTTP – Hypertext Transfer Protocol

IKM – Initial Keying Material

IP – Internet Protocol

IoT – Internet of Things

IPFS – InterPlanetary File System

IPNS – InterPlanetary Name System

IV – Inititalization Vector

JSON – JavaScript Object Notation

LIFO – Last In First Out

MAC – Message Authentication Codes

MAM – Masked Authenticated Messaging

MCMC – Monte Carlo Markov Chain

MHT – Merkle Hash Tree

MSP – Membership Service Provider

MSS – Merkle Signature Scheme

NAT – Network Address Translation

NFT – Non-Fungible Token

OSN – Ordering Service Nodes

OTS – One-Time Signatures

P2P – Peer to Peer

P2PK – Pay-to-PublicKey

P2PKH – Pay-to-PublicKeyHash

P2SH – Pay-to-ScriptHash

PBFT – Practical Bysantine Fault Tolerance

PoB – Proof of Burn

PoL – Proof of Luck

PoS – Proof of Stake

PoW – Proof of Work

PTM – Peer Transaction Manager

RPC – Remote Procedure Call

SBFT – Simplified Byzantine Fault Tolerance

SDK – Software Development Kit

SHA – Secure Hash Algorithm

SIV – Synthetic Initialization Vector

SPoA – Succint Proof of Access

SPoRes – Succint Proofs of Replications

TCP – Transmission Control Protocol

TEE – Trusted Execution Environment

TSA – Tip Selection Algorithm

VDF – Verifiable Delay Function

VSCC – Validation System ChainCode

WOTS – Winternitz One-Time Signature

WRE – Walka Radioelektroniczna

WWW – World Wide Web

1. Wstęp.

Technologia blockchain[1], kojarzona początkowo jedynie z dostarczaniem mechanizmów płatności z wykorzystaniem tzw. kryptowalut przeżywa w ostatnich latach intensywny rozwój, efektem czego jest odkrycie nowych obszarów zastosowania. Wraz z rozwojem tej technologii wprowadzono możliwość wykonywania obliczeń wewnątrz sieci blockchain. Niestety z upływem czasu pełna transparentność transakcji i zawartości blockchain spowodowała brak możliwości realizacji bardziej wymagających przypadków użycia, w szczególności tych związanych z przepływem danych, które nie powinny zostać upublicznione. Doskonałym przykładem może być tu aktualny rynek NFT (patrz rozdział 4.5), który w całości jest oparty o publicznie dostępne dane cyfrowe, jednocześnie nie dając ich właścicielowi żadnej możliwości czerpania zysków z wykorzystywania posiadanych przez niego dóbr i praw.

Wymiana informacji z wykorzystaniem klasycznych technologii informacyjnych wymaga istnienia zaufanych pośredników, potwierdzających tożsamość serwerów biorących udział w transmisji informacji. Przykładem może tu być wykorzystanie Centrów Certyfikacji (ang. CA – Certificate Authority) do poświadczania tożsamości serwera wykorzystywanego do komunikacji. Użytkownicy sieci zobowiązani są do zaakceptowania zastosowanych przez dostawcę certyfikatu technik identyfikacji, jednocześnie biorąc jedynie bierny udział w całym procesie. Przykładem nadużycia takiego zaufania jest afera DigiNotar z 2011 roku, kiedy to niezidentyfikowana grupa przestępcza dokonała fałszerstwa certyfikatów wystawionych m.in. dla irańskiej domeny Google, tym samym przechwytując ruch użytkowników kierowanych na te serwery. Podobne przypadki występowały także później, m.in. w roku 2015, gdy China Internet Network Information Center wystawiła jednemu ze swoich klientów certyfikat pośredni, pozwalający na dekryptaż całego ruchu kierowanego pomiędzy użytkownikami a witrynami internetowymi.

Zastosowanie centralnej infrastruktury pozwala na proste wprowadzenie cenzury i tym samym ograniczenie wolności słowa. Przykładem takich działań może być ograniczenie możliwości rozwiązywania adresów IP na podstawie wpisów DNS bądź ich całkowita eliminacja. Cenzura może także obejmować ograniczenie w dostępie do wybranej adresacji lub wyłączenie serwerów usługodawcy, przechowujących treść objętych cenzurą. W przypadku

[1] Blockchain, czyli łańcuch bloków, to technologia polegająca na tworzeniu ciągłej sekwencji wpisów, nazywanych blokami. Każdy z tych bloków jest chronologicznie powiązany z poprzednimi za pomocą metod kryptograficznych.

zastosowania technologii zdecentralizowanych, żadne z przedstawionych działań nie mogłoby mieć miejsca.

W sytuacji narastającego zagrożenia działaniami hybrydowymi oraz nakierowanymi na infrastrukturę krytyczną, istotne staje się zapewnienie bezpieczeństwa systemów i danych instytucji publicznych, prywatnych oraz obywateli. Zastosowanie technologii rejestru rozproszonego pozwala na decentralizację systemów kluczowych dla podejmowania decyzji mających bezpośrednie przełożenie na poziom bezpieczeństwa usług, systemów teleinformatycznych czy też skuteczność na polu walki. Technologia blockchain pozwala na decentralizację ośrodków decyzyjnych, co umożliwia znaczącą redukcję możliwych zagrożeń zewnętrznych i wewnętrznych.

Rozprawa doktorska jest podzielona na dziesięć głównych rozdziałów, które szczegółowo omawiają ewolucję i zastosowanie technologii blockchain. Rozdział drugi definiuje cel i główną tezę pracy, przedstawiając jednocześnie motywację stojącą za przeprowadzonymi badaniami. W rozdziale trzecim, czytelnik znajdzie szczegółowy opis technologii blockchain pierwszej generacji, z naciskiem na jej historyczne początki, najbardziej znaczącą implementację w postaci sieci Bitcoin, oraz omówienie kluczowych algorytmów konsensusu.

Rozdział czwarty przedstawia kolejne generacje sieci blockchain, określane jako 2.0 oraz 3.0. Pierwsze dwa podrozdziały zawierają omówienie sieci Ethereum oraz IOTA. W kolejnym podrozdziale przedstawiona została specyfikacja blockchain Hyperledger Fabric, dedykowanego zastosowaniom korporacyjnym. Ostatni podrozdział przedstawia nowoczesne technologie zdecentralizowanego przechowywania danych, takie jak IPFS i Arweave.

Piąty rozdział eksploruje praktyczne zastosowania technologii DLT (Distributed Ledger Technology) w zapewnianiu integralności, dostępności danych oraz w realizacji obliczeń rozproszonych, **podkreślając autorskie koncepcje wykorzystania tych technologii**.

Rozdział szósty zawiera szczegółowy opis opracowanego schematu uwierzytelnionego dostępu w środowisku rozproszonym. Pierwszy podrozdział określa możliwości przechowywania sekretów w sieci blockchain oraz wykorzystywania ich przez właściwe smart kontrakty. Drugi podrozdział określa kompletny schemat autoryzacji z wykorzystaniem smart kontrakt, wraz z fragmentami odpowiednich kodów źródłowych dla komponentów schematu. Trzeci podrozdział przedstawia wyniki ewaluacji schematu w kontekście przyjętych wymagań funkcjonalnych i niefunkcjonalnych.

Rozdział siódmy podsumowuje osiągnięcia pracy, wskazując na jej wkład naukowy i potencjalne implikacje dla przyszłych badań oraz rozwoju technologii. Ostatnie trzy rozdziały zawierają bibliografię, spis tabel oraz rysunków, które uzupełniają i wzbogacają treść rozprawy.

2. Cel i teza rozprawy.

Przedmiotem badań w niniejszej dysertacji naukowej jest zastosowanie zaawansowanych systemów blockchain do wzmacniania odporności systemów informatycznych[2], zarówno tych wykorzystywanych komercyjnie, jak i w szeroko pojętej infrastrukturze krytycznej. Badania mają charakter teoretyczno-praktyczny, co ma na celu przedstawienie schematów działania wykorzystywanych przez cyberprzestępców oraz metod ich mitygacji w kontekście ochrony interesu jednostki i interesu publicznego. **W wyniku przeprowadzonych badań zaproponowany został schemat, pozwalający na zastosowanie aktualnie istniejących technologii blockchain do realizacji uwierzytelnionego dostępu do danych w sposób całkowicie zdecentralizowany, tj. bez wykorzystania jakiejkolwiek zaufanej trzeciej strony.** Całość wykonywanych operacji zostanie przeprowadzona przez uczestników sieci, co uniemożliwi usunięcie dostępu do danych czy też wpłynięcie w sposób bezpośredni na proces udostępniania – np. poprzez wyłączenie serwerów usługodawcy. Tak zaprojektowany schemat będzie mógł być z powodzeniem wykorzystywany we wszelkich obszarach, gdzie krytycznym jest zapewnienie dostępności do danych, w sposób zapewniający uwierzytelnienie użytkownika, np. w sektorze publicznym dla zapewnienia niezawodnego dostępu do danych wrażliwych, takich jak dane medyczne czy dane osobowe obywateli.

Celem poznawczym dysertacji jest analiza potrzeb w zakresie zastosowanie technologii rejestru rozproszonego oraz decentralizacji usług w kontekście zapewnienia bezpieczeństwa jednostki, jak i organizacji.

Celem praktycznym jest zaproponowanie schematu realizacji uwierzytelnionego dostępu do danych przechowywanych w sposób zdecentralizowany, pomijający konieczność istnienia zaufanej trzeciej strony oraz wykluczający możliwość wyłączenia dostępu do usługi w sytuacjach kryzysowych.

Ogólny problem badawczy brzmi: *Jakie kierunki rozwoju i zastosowania technologii blockchain pozwolą na zwiększenie odporności systemów informatycznych*?

W celu rozwiązania powyższego problemu sformułowano szczegółowe problemy badawcze:

[2] Odporność systemu informatycznego to zdolność do utrzymania działania i zachowania bezpieczeństwa danych mimo zagrożeń takich jak awarie, ataki cybernetyczne, czy katastrofy naturalne.

1. *Jakie są możliwie technologie uwierzytelniania użytkowników, w przypadku braku zaufanej jednostki pośredniczącej?*
2. *Czy istnieje możliwość przechowywania danych szczególnie istotnych dla ochrony interesu państwa w sposób zdecentralizowany?*
3. *Jakie istnieją możliwości funkcjonowania sieci teleinformatycznej w przypadku blokady ruchu internetowego na poziomie międzynarodowym?*
4. *Jakie są możliwości prowadzenia komunikacji w środowisku o ograniczonym zaufaniu, w tym bez dostępu do sieci publicznej?*
5. *Jakie istnieją możliwości techniczne dla podejmowania autonomicznych decyzji przez uczestników niezaufanej sieci?*

Główna hipoteza badawcza:

Zastosowanie technologii blockchain pozwala na realizację uwierzytelnionego dostępu do danych zdecentralizowanych.

Szczegółowe hipotezy badawcze:

1. *Zastosowanie połączenia różnych technologii blockchain pozwala na stworzenie rozproszonej usługi szyfrowania danych.*
2. *Technologia blockchain pozwala na prowadzenie komunikacji w niezaufanym środowisku.*
3. *Możliwe jest utworzenie schematu składowania danych w sieci publicznej, gwarantującego integralność i poufność treści.*

Przedstawione hipotezy zostały zbadane na podstawie wyników przeprowadzonych badań oraz wykonanych implementacji. **Skonstruowane w ramach niniejszej rozprawy algorytmy mogą zostać z powodzeniem wykorzystane w procesie budowy nowych, a także modernizacji istniejących systemów informatycznych, pozwalając na zwiększenie poziomu ich odporności**.

3. Technologia blockchain 1.0

Rozdział ten skupia się na **pierwszej generacji technologii blockchain**, związanej głównie z aplikacjami finansowymi. Zawarty w niniejszym rozdziale rys historyczny opisuje początki koncepcji technologii blockchain aż po pierwsze implementacje. Zawarte w niniejszym rozdziale omówienie pierwszej i najbardziej znanej kryptowaluty – Bitcoin wprowadza czytelnika w szczegóły techniczne tego rozwiązania. Trzeci podrozdział koncentruje się na algorytmach konsensusu, stanowiących podstawę dla działania i bezpieczeństwa sieci blockchain. W szczególności wyjaśnione zostały podstawowe metody osiągania konsensusu w sieci takie jak Proof of Work, czy też problem bizantyńskich generałów oraz ich znaczenie w utrzymaniu integralności i niezmienności blockchain.

3.1. Rys historyczny

Pierwsza praca [1] S. Habera i W. S. Stornetty przedstawiająca koncepcję wykorzystania kryptografii do zabezpieczenia przed fałszerstwami jest datowana na rok 1993. Powszechnie uważa się, że przedstawione w niej koncepcje stanowiły fundament dla późniejszej technologii blockchain. Koncepcja ta została następnie rozszerzona w pracy W. Dai [2], datowanej na rok 1998. Opisywany projekt B-money nie został jednak nigdy w pełni zrealizowany.

Początki technologii blockchain, jaka jest znana dziś, są datowane na rok 2008, kiedy to osoba lub grupa osób identyfikująca się jako Satoshi Nakamoto opublikowała prace „Bitcoin: A Peer-to-Peer Electronic Cash System" [3][4]. Praca przedstawiała system, który może być wykorzystany do przeprowadzenia transakcji płatniczych pomiędzy dwoma jego użytkownikami, bez potrzeby wykorzystania trzeciej strony pośredniczącej w dokonywanej transakcji. Podstawowym założeniem było stworzenie systemu płatniczego, w którym bezpieczeństwo transakcji jest oparte jedynie o **kryptografię**, nie o **zaufanie**.

Praca [3] stanowiła przełom w technologii rejestrów rozproszonych, ze względu na zaprezentowanie rozwiązania problemu „podwójnego wydawania" środków. Problem ten dotyczył większości opracowywanych rozwiązań pieniądza cyfrowego, co powodowało, że środki mogły być w łatwy sposób pomnażane i wydawane więcej niż jeden raz. W wyniku tego zagrożenia, technologia ta nie mogła zostać szeroko wykorzystywana do transferu środków pomiędzy nieufającymi sobie stronami. Zaproponowane przez Nakamoto rozwiązanie tego problemu zakłada powiązanie każdej transakcji z transakcjami je poprzedzającymi. W tym celu stosowany jest skrót kryptograficzny, którego modyfikacja jest obliczeniowo niemożliwa. Powiązane ze sobą transakcje stanowią rejestr, który może być następnie walidowany przez

użytkowników sieci w celu zweryfikowania historii transakcji dla danego adresu i potwierdzenia, że posiada on wystarczające środki do przeprowadzenia transakcji.

Powszechnie przyjęta nazwa rejestru transakcji – blockchain wywodzi się z charakteru konstrukcji tego specyficznego typu bazy danych. Transakcje, cechujące się unikalnym numerem referencyjnym, znacznikiem czasowym oraz metadanymi są grupowane w bloki, a następnie w bloku są ze sobą wiązane z wykorzystaniem skrótów kryptograficznych tworząc bazę przechowywaną na wielu komputerach, nazywanych węzłami sieci (ang. nodes). Każdy z węzłów przechowuje pełen stan bazy danych, identyczny ze stanem przechowywanym na pozostałych węzłach. Odtworzenie aktualnego stanu sieci wymaga od węzła przeglądu wszystkich bloków, od bloku najmłodszego aż do bloku początkowego (ang. genesis block).

Wraz ze wzrostem zainteresowania technologią blockchain i Bitcoin, jako jej prekursorem i jednocześnie największym reprezentantem, nastąpił gwałtowny rozwój idei na wykorzystanie technologii rejestru rozproszonego. Poza początkowym zastosowaniem polegającym na transferze środków bez wykorzystania zaufanej trzeciej strony, w sposób gwarantujący pseudoanonimowość stron transakcji pojawiła się koncepcja stworzenia programowalnej logiki bazującej na danych zgromadzonych w blockchain. Jednym z ciekawszych projektów, był zaprezentowany w 2013 Ethereum [5]. Celem tego projektu było stworzenie zdecentralizowanej platformy, która pozwala na wykonywanie kodu smart kontraktów. Pozwala to na obsługę złożonych przypadków użycia, z wykorzystaniem logiki zaimplementowanej przez programistów. Może to być np. tworzenie nowych rynków, tokenów, przechowywanie rejestrów długów i zobowiązań, czy też przenoszenie środków zgodnie z instrukcjami, które zostały zaimplementowane w przeszłości i są publicznie znane. W porównaniu do Bitcoin, Ethereum tworzy system operacyjny, oparty o blockchain, który pozwala na obsługę dowolnych przypadków użycia opracowanych przez jego użytkowników. Był to jednocześnie pierwszy projekt, który pozwolił na obsługę przypadków użycia innych niż transfer środków.

Kolejne lata przyniosły skokowy rozwój technologii związanych z blockchain, w szczególności koncentrujący się na dalszym zwiększaniu efektywności tej technologii i przetwarzaniu coraz większych zbiorów danych. Na szczególną uwagę zasługują tu projekty rozwijające możliwości głównych blockchainów, takie jak Lightning Network dla Bitcoina [8], czy rollups dla Ethereum [9], a także projekty takie jak Solana [10], w których główny nacisk został położony na maksymalizację wydajności przy jednoczesnym obniżeniu kosztów obsługi sieci. Wzrastająca popularność idei zastosowania łańcuchów rozproszonych zaowocowała

także stworzeniem rozwiązań przeznaczonych stricte dla przedsiębiorstw. Niekwestionowanym liderem pozostaje tutaj opracowany w 2016 roku Hyperledger Fabric [11].

3.2. Bitcoin

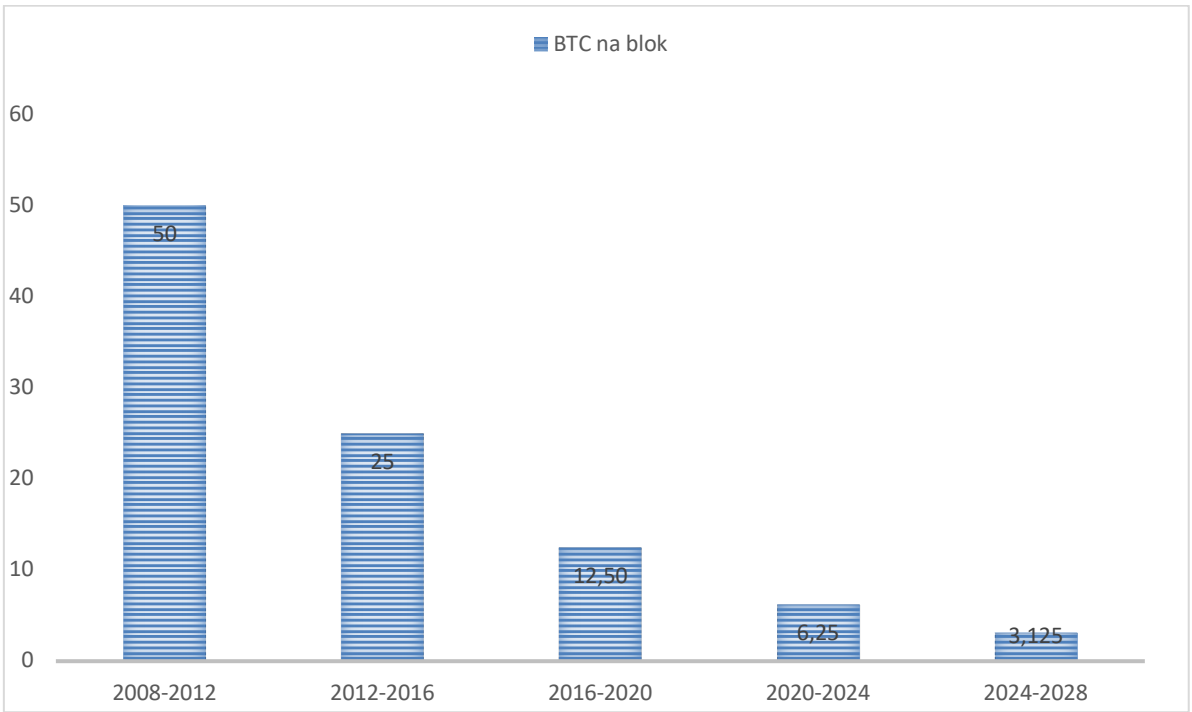
W niniejszym podrozdziale zamieszczony został dokładny opis technicznych aspektów działania sieci Bitcoin. Omówiona została struktura transakcji, proces tworzenia nowych środków oraz metody zabezpieczania transakcji.

Definicja 3.1.

Bitcoin (BTC) [3] to zdecentralizowana waluta cyfrowa pozwalająca na wykonywanie szybkich płatności pomiędzy użytkownikami sieci. Ze względu na brak centralnej instytucji odpowiadającej za rozliczanie transakcji, transfery mogą być wykonywane bez żadnych ograniczeń geograficznych ani legislacyjnych.

Bitcoin jako kryptowaluta został stworzony do wykonywania dwóch operacji:

- Zarządzania transakcjami, poprzez realizację transferu środków pomiędzy użytkownikami sieci
- Kreowania środków z wykorzystaniem procesu tzw. kopania zapewniając przy tym cechy ekonomiczne Bitcoin podobne jak w przypadku walut fiducjarnych[3].



Rysunek 1 Wartość nagrody za potwierdzenie bloku w sieci Bitcoin

[3] Waluta fiducjarna to rodzaj pieniądza, który nie ma wewnętrznej wartości i nie jest wspierany przez żadne fizyczne dobra, takie jak złoto czy srebro. Jego wartość opiera się wyłącznie na zaufaniu i akceptacji użytkowników oraz gwarancji państwa, które go emituje.

Całkowita liczba możliwych do wykreowania w procesie kopania Bitcoinów wynosi 21 milionów. W procesie wydobywania bitcoinów, górnicy[4] potwierdzają transakcje sieci rozwiązując zadanie polegające na odnalezieniu wartości heksadecymalnej, która w połączeniu z transakcjami danego bloku wygeneruje wartość skrótu kryptograficznego mniejszą od zadanej – poziomu trudności[5]. Oczekiwany czas wydobycia bloku, czyli czas rozwiązania tej zagadki jest określony na 10 minut. Sieć automatycznie dostosowuje trudność wydobycia bloków co 2016 poprawnie wygenerowanych bloków (14 dni), tak aby zachować oczekiwaną wartość czasu potwierdzania bloku. Początkowo, nagroda za wydobycie bloku wynosiła 50 BTC, jednakże wraz z każdą wielokrotnością wydobytych 210000 bloków wartość ta jest obniżana o połowę aż do wydobycia wszystkich możliwych BTC. Szacuje się, że ostatni BTC zostanie wydobyty w roku 2140.

W Bitcoin wyróżniamy dwa rodzaje transakcji:

- transakcje tworzenia środków (ang. coinbase),
- transakcje regularne.

Obydwa rodzaje transakcji są oparte o taką samą architekturę i działają w podobny sposób. Każda z nich ma jednak inny cel – transakcje regularne są wykorzystywane do transferu środków pomiędzy użytkownikami sieci, podczas gdy transakcje tworzenia środków są wykorzystywane do tworzenia nowych BTC i wprowadzania ich do obiegu.

Kolejne transakcje regularne są tworzone w oparciu o uprzednio istniejące transakcje, tworząc tym samym łańcuch powiązań. Każda transakcja wykorzystuje wyjście poprzedniej transakcji jako swoje wejście. Transakcje są reprezentowane jako ciąg bitowy i mogą być przedstawione z wykorzystaniem następującej struktury [13]:

Tabela 1 Struktura danych transakcji sieci Bitcoin [13]

Pole	Typ (rozmiar)	Opis
`nVersion`	`int (4 bajty)`	Wersja transakcji.

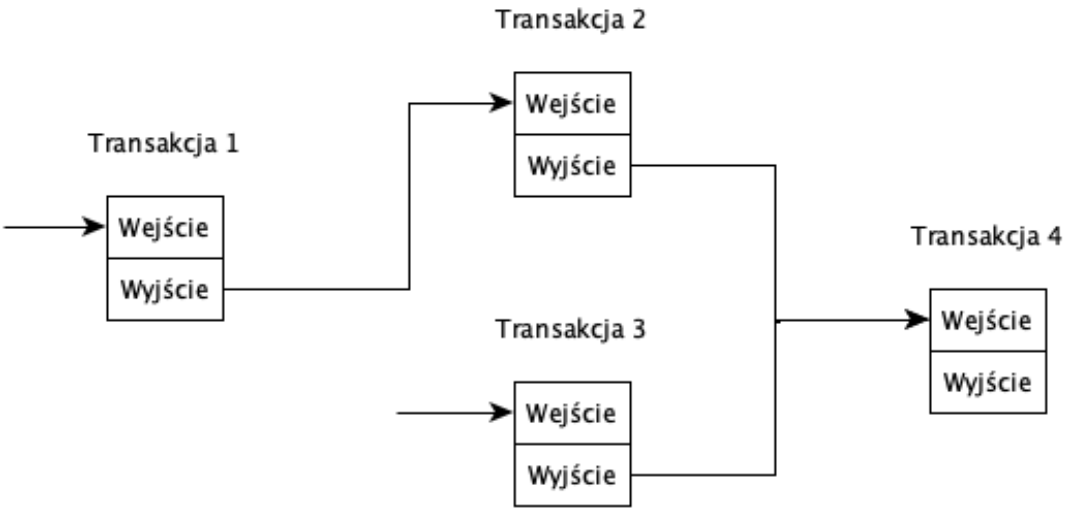
[4] Górnik sieci Bitcoin to uczestnik sieci, który używa mocy obliczeniowej swojego sprzętu komputerowego do rozwiązywania skomplikowanych zadań matematycznych, które umożliwiają potwierdzanie transakcji i dodawanie nowych bloków do blockchain.

[5] Poziom trudności w kontekście jest parametrem określającym minimalną wartość, jaką musi osiągnąć skrót kryptograficzny bloku, aby został zaakceptowany przez sieć.

#vin		VarInt (1-9 bajtów)	Liczba wpisów na liście wejść transakcji.
vin []	hash	uint256 (32 bajty)	Skrót SHA-256 poprzedzającej transakcji. Wykorzystywany jako unikalny identyfikator.
	n	uint (4 bajty)	Indeks oczekiwanego wyniku transakcji.
	scriptSigLen	VarInt (1-9 bajtów)	Długość pola scriptSig wyrażona w bajtach.
	scriptSig	CScript (zmienna)	Rezultat skryptu poprzedniej transakcji.
	nSequence	uint (4 bajty)	Numer sekwencyjny wejścia transakcji.
#vout		VarInt (1-9 bajtów)	Liczba wpisów w tablicy transakcji wyjściowych.
vout []	nValue	int64_t (8 bajtów)	Liczba BTC w transakcji, wyrażana w jednostkach równych 10^{-8} BTC, określonych jako satoshi.
	scriptPubKeyLen	VarInt (1-9 bajtów)	Długość pola scriptPubKey
	scriptPubKey	CScript (zmienna)	Skrypt określający warunki, jakie muszą być spełnione, żeby możliwe było wydanie powiązanych BTC.
nLockTime		uint (4 bajty)	Znacznik czasowy określający docelowy blok.

Każda transakcja zawiera listę wartości wejściowych i wyjściowych. Wejścia pewnej transakcji stanowią wyjścia transakcji wykonanych uprzednio. Powyższe pozwala na stworzenie łańcucha transakcji, w którym każdy kolejny wpis jest zależny od transakcji przeprowadzonych wcześniej. Ta właściwość w znaczący sposób utrudnia możliwość zmiany danych zawartych w transakcjach. Poprawna walidacja transakcji oznacza, że proces weryfikacji potwierdza, iż każda transakcja spełnia ustalone zasady sieci: wejścia transakcji nie zostały wcześniej wykorzystane (zapobiega to podwójnemu wydawaniu), a cały łańcuch

transakcji jest spójny i niezmieniony. W miarę upływu czasu i dodawania nowych transakcji, coraz więcej zapisów w rejestrze musi być sprawdzanych, co zapewnia bezpieczeństwo i integralność całego systemu blockchain.



Rysunek 2 Łańcuch transakcji

W przedstawionej w tabeli 1. strukturze należy wyróżnić tablice `vin[]` oraz `vout[]`. Pierwsza z nich określa wyjścia poprzednich transakcji, które są identyfikowane poprzez pary `(hash, n)`.

Pole `hash` jest unikalnym identyfikatorem transakcji, podczas gdy pole `n` określa indeks wyjścia tej transakcji. Aby możliwe było przypisanie środków BTC w danej transakcji, pole `scriptSig` musi zawierać poprawną wartość dla `scriptPubKey` wybranego wyjścia.

Tablica `vout[]` reprezentuje wyjścia transakcji. Każda transakcja musi posiadać co najmniej jedno wyjście. Liczba transferowanych w ramach transakcji środków jest określana w polu `nValue`, a jego wartość oznacza ilość transferowanych tokenów[6] wyrażoną jako wielokrotność 10^{-8} BTC.

Pole `scriptPubKey` jest wykorzystywane do określenia odbiorcy przesyłanych środków, dla każdego z wyjść transakcji. Ostatnim z pól transakcji jest `nLockTime`, którego wartości jest interpretowana jako znacznik czasu lub numer bloku, po którym transakcja będzie mogła być ujęta w bloku. Zgodnie z pracą [13] wartości posiadają następujące znaczenie:

[6] Token to najmniejsza kwantyfikowalna jednostka wartości w sieci, która umożliwia przeprowadzanie transakcji finansowych.

- 0 – transakcja niezablokowana
- <500,000,000 – numer bloku, po którym transakcja może zostać ujęta w bloku
- ≥500,000,000 - znacznik czasowy UNIX po którym transakcja może zostać ujęta w bloku

Transakcje tworzenia środków wykorzystują taką samą strukturę danych jak transakcje regularne, jednak niektóre z pól są wykorzystywane do przechowywania szczególnych wartości:

- `#vin` – ponieważ w tej transakcji kreowane są nowe środki, nie wymaga się wprowadzenie transakcji wejściowej. Wartość `#vin` jest zawsze ustawiana na 1.
- `vin[]` – wykorzystywana jest jedna stała transakcja wejściowa, która nie stanowi transakcji poprzedniej. Identyfikator jest ustawiany na wartość domyślną `(hash, n) =(0,`2^{32}`-1).`
- `scriptSigLen` – w przypadku transakcji tworzenia środków pole to jest też nazywane `coinbaseLen`, przechowuje długość pola `scriptSig`.
- `scriptSig` – pole określane również jako `coinbase` przechowujące numer bloku oraz inne niezbędne dane potrzebne do rozwiązania problemu konsensusu.
- `nValue` – pole określające wartość przyznanej nagrody za wydobycie bloku.

Dane transakcji podlegają poniższym ograniczeniom:

- Całkowity rozmiar danych transakcji nie może przekraczać 10000 bajtów.
- Dane scriptSig dla każdego z wejść transakcji nie może przekraczać 500 bajtów.
- Nie można realizować transakcji, w których wartość wnoszonych opłat transakcyjnych jest większa od 1/3 sumy przesyłanych środków.
- Dla transakcji tworzenia środków, środki pozostają zablokowane przez następne 100 bloków, aby uniknąć wydawania środków z odrzucanych rozgałęzień głównego łańcucha sieci.[7]

[7] Główny łańcuch sieci Bitcoin, znany również jako główny blockchain, to ciągła, chronologiczna sekwencja bloków zawierających transakcje, która jest uznawana za oficjalną wersję historii transakcji w sieci Bitcoin.

Protokół Bitcoin wykorzystuje niekompletny w sensie Turinga[8] język Script. Język ten został stworzony specjalnie na potrzeby protokołu, nie pozwala na realizację obliczeń ani na rozwiązywanie złożonych problemów. Na język składa się zestaw OP_CODES, które stanowią zarezerwowane słowa kluczowe odpowiadające za sposób interpretacji zapisu w skrypcie. W protokole Bitcoin skrypty są wykorzystywane w procesie odbierania przetransferowanych środków i stanowią część transakcji w sieci. Wyróżniane są dwie podstawowe grupy skryptów – skrypty wyzwania i skrypty odpowiedzi. Wyzwania są zawarte w każdym z wyjść transakcji i zawierają wyzwanie, które musi być wypełnione przez odbiorcę w celu uwolnienia środków. Jest to określenie warunków, na jakich dane środki mogą zostać zwolnione. Skrypty odpowiedzi są zawarte w wejściach transakcji i zawierają rozwiązanie lub odpowiedź na wyzwanie do transakcji, w której były zawarte w pracy [13]. Celem stosowania skryptów jest ograniczenie dostępu do środków dla nieautoryzowanych stron, poprzez stosowanie wyzwania, które niskim nakładem może zostać rozwiązane przez adresata, przy jednoczesnym braku obliczeniowej możliwości jego rozwiązania przez stronę nieuprawnioną.

Węzły sieci Bitcoin obsługują ograniczoną liczbę szablonów skryptów[9], wśród których należy wymienić:

- Pay-to-PubKey
- Pay-to-PubKeyHash
- Pay-to-ScriptHash
- Multisig
- Nulldata

Pay-to-PubKey (P2PK) pozwala na transfer środków bezpośrednio do posiadacza klucza publicznego. Aby odebrać środki należy potwierdzić posiadanie dostępu do odpowiadającego mu klucza prywatnego poprzez dostarczenie prawidłowego podpisu cyfrowego dołączonego do skryptu odpowiedzi [13]. Ten szablon jest uznawany obecnie za przestarzały, jednakże wciąż akceptowany.

Pay-to-PubKeyHash (P2PKH) to rozwinięcie szablonu P2PK, w którym klucz publiczny został zastąpiony jego skrótem kryptograficznym. Pozwala to na zmniejszenie rozmiaru

[8] Językiem niekompletnym w sensie Turinga nazywany jest język programowania, który nie jest w stanie symulować dowolnej maszyny Turinga, a tym samym nie może wykonać wszystkich możliwych obliczeń.

[9] Szablon skryptu to zdefiniowany format skryptu transakcyjnego, który określa sposób, w jaki transakcje są strukturyzowane i weryfikowane przez węzły sieci.

transakcji, co jednocześnie przy narzucanych przez protokół ograniczeniach na jej rozmiar pozwala na zwiększenie poziomu złożoności implementowanej logiki. Dodatkowo zwiększana jest odporność transakcji na ataki wymierzone w klucz publiczny, ponieważ aby skutecznie odzyskać klucz prywatny ze skrótu, konieczne jest także odnalezienie właściwego klucza publicznego odpowiadającego temu skrótowi. Skrypt pozwala na zastosowanie kilku rodzajów funkcji skrótu. Przykładami mogą tu być:

- OP_SHA256 – oznacza wykorzystanie SHA-256 w procesie obliczania skrótu.
- OP_HASH160 – wykorzystywane są dwa algorytmy:
  - pierwszy skrót jest obliczany z wykorzystaniem SHA-256,
  - docelowy skrót jest obliczany ze skrótu pierwszego z wykorzystaniem algorytmu RIPEMD_160.
- OP_HASH256 – skrót jest obliczany z wykorzystaniem konkatenacji skrótów SHA-256. Pierwszy skrót jest obliczany na podstawie klucza publicznego, docelowy na podstawie skrótu pierwszego.

Pay-to-ScriptHash (P2SH) przypomina skrypt P2PKH, jednakże transfer nie jest dokonywany do posiadacza danego klucza publicznego, lecz do posiadacza skryptu, nazywanego także adresem Pay-to-ScriptHash. Nadawca takiej transakcji tworzy skrypt wyzwania, ale zamiast dołączać jego kod do transakcji, dołącza jego skrót kryptograficzny. Aby odebrać transferowane środki, odbiorca musi przedstawić skrypt, którego skrót odpowiada wartości skrótu zapisanej w transakcji, a także wszystkie inne dane wejściowe które są niezbędne do prawidłowego jego wykonania.

Multisig to szablon skryptu, który pozwala nadawcy na przesłanie środków na adres zabezpieczonych przez więcej niż jeden klucz publiczny. Skrypt wyzwania[10] zawiera listę *n* kluczy publicznych oraz liczbę *m* (mniejszą bądź równą *n*), która określa minimalną liczbę podpisów wymaganych do autoryzacji transakcji i zwolnienia środków. Odbiorca musi dostarczyć co najmniej *m* ważnych podpisów cyfrowych, odpowiadających kluczom publicznym wymienionym w skrypcie wyzwania, aby transakcja mogła zostać zatwierdzona.

Multisig może zostać zrealizowany z wykorzystaniem szablonów P2PKH lub P2SH. Ten rodzaj transakcji jest wykorzystywany wtedy, gdy transfer ma być dokonany do grupy

[10] Skrypt wyzwania (ang. challenge Script) w sieci Bitcoin to fragment skryptu transakcyjnego określający zestaw kryptograficznych warunków, które muszą być spełnione przez prezentującego transakcję (odbiorcę środków), w tym weryfikację odpowiednich podpisów cyfrowych i kluczy publicznych, aby umożliwić zwolnienie i transfer zablokowanych środków.

właścicieli, przy założeniu, iż żaden z indywidualnych członków grupy nie może uzyskać dostępu do środków.

Nulldata to skrypt, który nie narzuca żadnych wymagań odnośnie do skryptu wyzwania, a co za tym idzie nie stawia żadnych wymagań na odpowiadający mu skrypt odpowiedzi. Celem tego rodzaju skryptów jest nie transfer środków, lecz dołączenie danych do transakcji. Transakcje zawierające tego typu skrypt pozwalają na wykonanie transferu BTC w ilości 0, co oznacza, że nie podlegają ograniczeniom dotyczącym stosunku transferowanych BTC i opłaty transakcyjnej.

Transakcje, które zawierają skrypty wymagające potwierdzenia przez odbiorcę posiadania dostępu do odpowiedniego klucza prywatnego oczekują dostarczenia zgodnego podpisu cyfrowego dla przesłanych danych. Podpisy te wykorzystują różne typy obliczanych skrótów kryptograficznych dla danych, które mają zostać podpisane. Aktualnie protokół obsługuje następujące typy skrótów:

- SIGHASH_ALL
- SIGHASH_SINGLE
- SIGHASH_NONE
- SIGHASH_ANYONECANPAY

Domyślnie stosowany jest typ SIGHASH_ALL, dla którego wszystkie dane transakcji są wykorzystywane do utworzenia podpisu. Skrót jest obliczany następująco:

$$H = hash(nVersion||\#vin||\sum_{i=0}^{\#vin} vin[i]\,||\#vout||\sum_{j=0}^{\#vout} vout[j]||nLocktime$$

Dla SIGHASH_SINGLE uwzględniane jest tylko jedno wyjście transakcji, poza tym stosowany jest taki sam zestaw danych jak dla SIGHASH_ALL:

$$H(x) = hash(nVersion||\#vin||\sum_{i=0}^{\#vin} vin[i]\,||x+1||vout[x]||nLocktime$$

W przypadku SIGHASH_NONE wygenerowany skrót nie uwzględnia żadnych wyjść transakcji. Jest generowany zgodnie z poniższym wzorem:

$$H = hash(nVersion||\#vin||\sum_{i=0}^{\#vin} vin[i]\,||0||nLocktime$$

Dla SIGHASH_ANYONECANPAY uwzględniane jest tylko aktualne wejście transakcji, pozostałe wartości są pobierane tak jak dla SIGHASH_ALL, a skrót jest generowany zgodnie z poniższym wzorem:

$$H(x) = hash(nVersion||1||vin[x]||\#vout||\sum_{j=0}^{\#vout} vout[j]||nLocktime$$

Wytworzenie podpisu dla zdefiniowanego typu skrótu jest możliwe po wykonaniu trzech kroków:

- W pierwszym kroku, typ podpisu jest dołączany do zmodyfikowanych danych transakcji.
- W kroku drugim dane te są szyfrowane z wykorzystaniem klucza prywatnego.
- W trzecim kroku dokonywane jest dołączenie ostatniego bajtu typu podpisu do zaszyfrowanych danych, tak aby odbiorca w łatwy sposób mógł określić jakie dane były podpisywane, bez potrzeby ich deszyfrowania.

Podpis jest generowany zgodnie z poniższym równaniem:

$$S = sign(transaction\_data||sigtype)||sigtype[last\_byte]$$

Adresy portfeli w sieci Bitcoin są generowane na podstawie skrótów, wykorzystywanych w skryptach P2PKH oraz P2SH [14]. Adres składa się z 34 znaków, zakodowanych z wykorzystaniem kodowania Base-58 [15]. Przykładem takiego adresu sieci BTC jest *1Lbcfr7sAHTD9CgdQo3HTMTkV8LK4ZnX7.* Proces generowania adresu składa się z poniższych czterech kroków:

1. Określenie numeru wersji

Dla adresu PubKeyHash wykorzystywany jest bajt 0x00, dla ScriptHash 0x05. Następnie te wartości są konwertowane przez koder do wartości 1 dla PubKeyHash lub 3 dla ScriptHash.

2. AddrHash

Wartość AddrHash jest kombinacją numeru wersji oraz skrótu klucza publicznego lub skryptu. Do wygenerowania 20 bajtowego skrótu wykorzystywane są dwie funkcje skrótu – SHA-256 oraz RIPEMD-160, zgodnie z poniższymi wzorami:

$$AddrHash = RIPEMD\_160(SHA\_256(script))$$

$$AddrHash = RIPEMD\_160(SHA\_256(PubKey))$$

3. Suma kontrolna

Adres zawiera sumę kontrolną, która pozwala na łatwe wykrywanie błędów przy wprowadzaniu adresu. Sumę kontrolną stanowią pierwsze cztery bajty dwukrotnie wykonanego skrótu SHA-256 na wartości AddrHash.

$$SK = SHA\_256(SHA\_256(AddrHash))[0-3]$$

4. Kodowanie adresu

Ostatnim krokiem jest zakodowanie adresu z wykorzystaniem kodowania Base-58. Jest ono przeprowadzane zgodnie z poniższym wzorem:

$$Adres = Base58\_encode(AddrHash||SK)$$

Transakcje w sieci Bitcoin są grupowane w bloki. Każdy z bloków ma dwa główne komponenty – nagłówek oraz dane. Nagłówek posiada stały rozmiar 80 bajtów, natomiast dane mają zmienny rozmiar i są zależne od ilości transakcji zawartych w bloku. Przyjmuje się, że górny limit wielkości bloku to 1 MB. Podobnie jak ma to miejsce w przypadku transakcji, skrót nagłówka bloku jest wykorzystywany jako identyfikator. W poniższej tabeli przedstawiona została struktura bloku w sieci Bitcoin.

Tabela 2 Struktura bloku Bitcoin [13].

Nazwa pola		Typ (rozmiar)	Opis
Nagłówek	`nVersion`	`int (4 bajty)`	Wersja bloku
	`HashPrevBlock`	`uint256 (32 bajty)`	Skrót nagłówka poprzedniego bloku
	`HashMerkleRoot`	`uint256 (32 bajty)`	Skrót korzenia drzewa skrótów, zawierający wszystkie transakcje zawarte w bloku
	`nTime`	`uint (4 bajty)`	Znacznik czasowy utworzenia bloku
	`nBits`	`uint (4 bajty)`	Wartość celu dla proof-of-work
	`nNonce`	`uint (4 bajty)`	Wartość losowa wykorzystywana do rozwiązania proof-of-work (patrz rozdział 3.3)

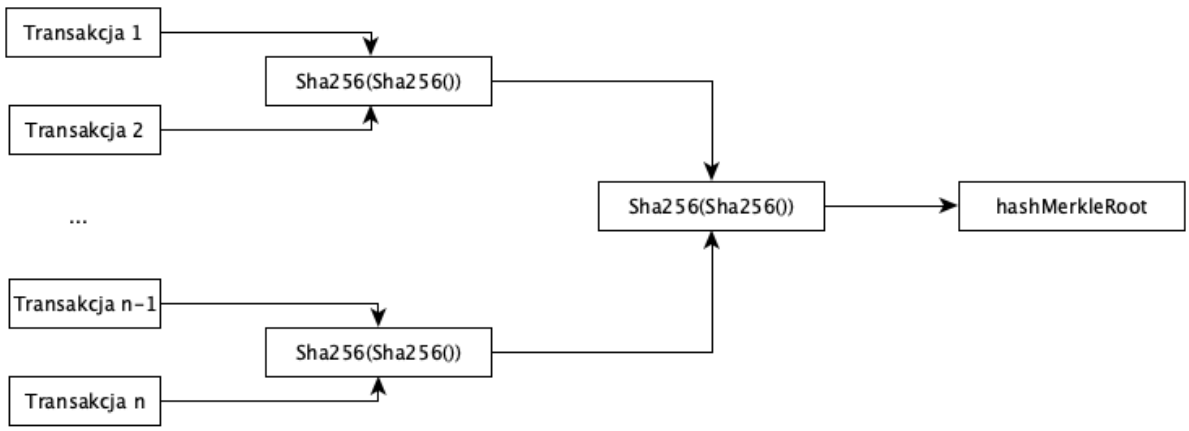
Dane	#vtx	VarInt (1-9 bajtów)	Liczba transakcji w tablicy vtx []
	vtx []	Transaction (zmienna)	Tablica wszystkich transakcji zawartych w bloku

Wartość pola nVersion jest równa 1 lub 2, w zależności od wykorzystywanej wersji struktury bloku. Inne wartości zawarte w tym polu powodują, że blok jest odrzucany. Pole HashPrevBlock zawiera skrót nagłówka poprzedzającego bloku, co pozwala na wiązanie ze sobą bloków sieci. Wartość ta jest obliczana zgodnie z poniższym wzorem:

$$HashPrevBlock = SHA\_256(SHA\_256(PrevHeader))$$

$$PrevHeader = nVersion||HashPrevBlock||HashMerkleRoot||nTime||nBits||nNonce$$

Pole HashMerkleRoot zostało wprowadzone w celu ograniczenia liczby danych przechowywanych w blockchain. W szczególności wykorzystywane jest to przez węzły sieci, które nie gromadzą całej historii transakcji, lecz skupiają się na analizie zadanego okresu. Dodatkowym atutem wprowadzenia tego pola było ułatwienie wykrywania naruszenia integralności transakcji zawartych w danym bloku.



Rysunek 3 Schemat obliczania hashMerkleRoot

Pole nTime zawiera znacznik czasowy w formacie UNIX reprezentujący czas utworzenia danego bloku. Wartość ta jest przypisywana przy rozpoczęciu tworzenia bloku, zatem będzie mniejsza niż rzeczywisty czas utworzenia bloku i jego rozpropagowania w sieci. W kolejnym polu nBits przechowywana jest skompresowana reprezentacja wartości docelowej do

rozwiązania proof-of-work (patrz rozdział 3.3.). Oryginalna 256-bitowa wartość jest kompresowana do wartości 32 bitowej z wykorzystaniem algorytmu [16], opisanego poniżej:

1. Przekształć wartość do base256, tak aby każdy bit był reprezentowany przez dwa symbole heksadecymalne.
2. Jeżeli najbardziej znaczący bit jest większy niż 127 (0x7f), dodaj 0 do pierwotnego numeru.
3. Pierwszy bit wyjścia powinien być reprezentowany przez liczbę bajtów w początkowej wartości. Jeżeli dodane zostało 0 w kroku 2., również należy je uwzględnić.
4. Ostatnim krokiem jest dodanie trzech najbardziej znaczących bajtów do symbolu obliczonego w kroku 3. W wyniku otrzymana zostanie wartość 4 bajtowa.

Odtworzenie początkowej wartości jest możliwe z wykorzystaniem następującej formuły:

$$n = h_1 h_2 h_3 \times 2^{8\times(h_0-3)}$$

gdzie h_i oznacza i-ty bajt wartości `nBits`, $i \in \{0,1,2,3\}$.

Pole `nNonce` zawiera cztery losowe bajty wykorzystywane do dodania losowości w procesie kopania waluty. W procesie tym należy tak dopasować wartość `nNonce`, aby docelowy obliczony skrót był mniejszy niż zadany przez aktualny poziom trudności sieci.

W polu `#vtx` przechowywana jest wartość liczbowa oznaczająca liczbę elementów w tablicy `vtx[]` - ilość transakcji przechowywanych w bloku. Liczba transakcji jest różna dla różnych bloków i ograniczona przez maksymalny rozmiar bloku.

Tak zdefiniowane bloki są wydobywane przez węzły (zwane również górnikami), które wykonują obliczenia mające na celu znalezienie wartości `nNonce`, która po obliczeniu wartości skrótu kryptograficznego, generuje skrót bloku mniejszy niż określony próg. Ten próg, zwany trudnością, jest reprezentowany przez 256-bitową wartość. Pierwotnie, w protokole Bitcoin, wartość trudności była ustawiona tak, że pierwsze 32 bity były zerami, a następne 224 bity były jedynkami, co stanowiło stosunkowo niski poziom trudności. Próg ten jest dynamicznie dostosowywany co 2016 bloków, co odpowiada okresowi około dwóch tygodni. Celem dostosowania jest utrzymanie średniego czasu między znalezieniem kolejnych bloków na poziomie około dziesięciu minut. Dzięki temu proces wydobycia jest stabilny, a nagroda za wydobycie bloku — nowo wytworzone BTC oraz opłaty transakcyjne zawarte w bloku — jest przypisywana adresowi górnika, który odnalazł odpowiednią wartość `nNonce`.

Węzły podczas generowania bloku pobierają transakcje do umieszczenia w danym bloku z zgodnie z ich priorytetem. Priorytet dla danej transakcji jest wyznaczany z wykorzystaniem następującej formuły:

$$P = \frac{\sum_{i=0}^{\#vin}(wartość(vin[i]) \times wiek(vin[i])}{długość}$$

Wartość jest określana jako liczba BTC zawartych w transakcji, wiek jest równy liczbie bloków wygenerowanych od czasu utworzenia transakcji, długość oznacza rozmiar danych transakcji wyrażony w bajtach.

Proces wydobycia nowego bloku wymaga wykonania czterech kroków:

- W pierwszym kroku należy zebrać wszystkie rozgłoszone transakcje i wybrać te, które mają zostać dołączone do bloku. Pierwsza transakcja dołączona do bloku to transakcja kreowania nowych środków, która jest tworzona przez górnika i określa sposób wypłaty środków za odnalezienie poprawnego bloku.
- W drugim kroku dokonywana jest walidacja poprawności transakcji zamieszczanych w bloku. Jeżeli transakcja nie zostanie pozytywnie zweryfikowana, to nie może być umieszczona w bloku.
- W kroku trzecim wybierany jest ostatni blok najdłuższego łańcucha bloków sieci. Blok ten będzie stanowił punkt odniesienia dla nowo tworzonego bloku. Obliczany jest skrót kryptograficzny nagłówka tego bloku, zgodnie z poniższym wzorem.

$$Hash = SHA\_256(SHA\_256(Header))$$

$$Header = nVersion||HashPrevBlock||HashMerkleRoot||nTime||nBits||nNonce$$

  Następnie jest on umieszczany w nagłówku generowanego bloku.
- W czwartym, ostatnim kroku rozwiązywany jest problem proof-of-work – wyznaczana jest wartość `nNonce`. Po odnalezieniu poprawnej wartości blok ten może zostać rozgłoszony do innych węzłów sieci.

Szczegółowy opis konsensusu *proof-of-work* został zawarty w kolejnym podrozdziale.

3.3. Algorytmy konsensusu

W niniejszym podrozdziale został zamieszczony opis podstawowych algorytmów konsensusu stosowanych w sieciach blockchain. Zdefiniowane zostały algorytmy konsensusu oparte o problem bizantyjskich generałów, a także wykorzystywane w publicznych blockchain konsensusy Proof of Work, Proof of Stake oraz ich modyfikacje.

Algorytmy konsensusu wywodzą się bezpośrednio z problemu bizantyjskich generałów, który po raz pierwszy został opisany przez Lamporta w pracy [18]. W problemie tym zakłada się spełnienie następujących warunków:

- Bizancjum to stolica Cesarstwa Bizantyńskiego.
- Bizancjum składa się z kilku lenn, z których każde ma swojego generała oraz podległych mu żołnierzy.
- Każdy generał może wydać dwa rodzaje rozkazów: atak lub wycofanie.
- Jedynie, gdy wszyscy generałowie wydadzą taki sam typ rozkazu – atak lub wycofanie – można zminimalizować straty i wygrać wojnę.
- Ze względu na rozległość Cesarstwa Bizantyńskiego nie jest możliwe przeprowadzenie bezpośredniej dyskusji pomiędzy generałami. Nie mogą oni opuszczać swojego lenna.
- Rozkazy wydawane przez generałów są dostarczane przez sygnalistów.
- Generałowie podejmują swoją ostateczną decyzję poprzez wysłanie swoich rozkazów do innych generałów i zebraniu ich rozkazów.
- Sygnaliści są uczciwi i nie modyfikują rozkazów.
- Niektórzy z generałów są zdrajcami, którzy mogą wysyłać fałszywe rozkazy lub wysyłać odmienne rozkazy do różnych generałów, tak aby zmienić ostatecznie podejmowaną decyzję.

Problem ten polega na uzyskaniu konsensusu co do podejmowanej decyzji wśród uczciwych generałów, nawet gdy część z generałów to zdrajcy.

Algorytm konsensusu stosowany w sieciach blockchain ma za zadanie zapewnić spójność danych w sytuacji, gdy rozproszony system posiada kilka niewłaściwie działających węzłów. Niewłaściwie działające węzły możemy przypisać do dwóch kategorii – węzły zatrzymane (Crash Fault Node, CFN) oraz węzły bizantyńskie (Bysantine Fault Node, BFN). Węzły CFN mogą ulec awarii, która polega na zatrzymaniu ich działania. Węzły te nie prowadzą żadnych złośliwych działań [19]. W przypadku takich węzłów, przesyłana wiadomość może być albo

opóźniona, albo całkowicie utracona. W przypadku węzłów BFN wiadomości mogą być wysyłane w sposób błędny, a także wysyłane w różnej postaci do różnych węzłów sieci, co ma na celu zaburzenie procesu osiągania konsensusu. Jeżeli w systemie występują jedynie węzły CFN, wtedy rozwiązaniem jest zastosowanie algorytmów dla systemów tolerujących awarie[11], np. Paxos [20,21]. Nie jest to rozwiązanie właściwe dla sieci Blockchain, ponieważ węzły tej sieci są kontrolowane przez osoby indywidulane i konsorcja, które mogą próbować nadużyć swojej roli w sieci, w celu osiągnięcia korzyści. Powyższe wskazuje na konieczność zastosowania algorytmu konsensusu, który jest odporny na występowanie BFN.

Twierdzenie FLP [22]

Żaden protokół konsensusu nie jest całkowicie poprawny w sytuacji wystąpienia pojedynczej awarii.

Konsensus dla sieci blockchain opiera się na twierdzeniu FLP [22]. Zgodnie z tym twierdzeniem, tak długo jak jeden z procesów jest nieresponsywny[12], tak długo nie jest możliwe osiągnięcie pełnego konsensusu w systemie rozproszonym wykorzystującym komunikację asynchroniczną.

W przypadku zastosowaniu komunikacji synchronicznej, wykorzystywany jest zegar o stałej częstotliwości dopuszczający występowanie błędów w ograniczonym czasie.

Dla modelu asynchronicznego nie ma możliwości wykorzystania mechanizmu wygasania błędów[13], ponieważ nie istnieje wspólny zegar. Większość stosowanych obecnie algorytmów konsensu wykorzystuje model komunikacji synchronicznej lub tzw. słaby model komunikacji synchronicznej, w którym zaimplementowany został mechanizm wygasania dla transmitowanych wiadomości. W sieci blockchain oznacza to, iż wiadomość może zostać opóźniona, jednakże w pewnym zadanym czasie powinna zostać dostarczona do odbiorcy. Jeżeli ten zadany czas ulegnie przekroczeniu, to zostanie przyjęte założenie, iż węzeł wysyłający wiadomość ma awarię. Zgodnie z powyższym wymaga się, aby algorytm konsensusu dla sieci blockchain gwarantował zachowanie zasad spójności[14] oraz żywotności.

[11] System tolerujący awarie (fault tolerant system) to system, który został zaprojektowany i skonstruowany w taki sposób, aby mógł kontynuować pracę nawet w przypadku wystąpienia błędów lub awarii jego komponentów.

[12] Nieresponsywność to stan, w którym dany proces lub system przestaje odpowiadać na zewnętrzne żądania lub działania w sposób oczekiwany lub w ogóle przestaje odpowiadać.

[13] Mechanizm wygasania błędów to technika stosowana w systemach komunikacji, która polega na ustanowieniu maksymalnego czasu oczekiwania na odpowiedź na przesłane żądanie.

[14] Spójność w kontekście blockchain oznacza, że wszystkie uczciwe węzły sieci przechowują jednolitą historię wpisów, eliminując wszelkie sprzeczności i zapewniając integralność danych.

Spójność w tym przypadku oznacza, iż jeżeli transakcja jest ważna na uczciwym węźle, jest także ważna na innych uczciwych węzłach sieci. Spójność pozwala na zapewnienie, że atak podwójnego wydawania środków [23] nie może mieć miejsca, jeżeli większość węzłów sieci stanowią węzły uczciwe. Żywotność oznacza, że wszystkie ważne transakcje wysłane przez uczciwe węzły sieci muszą zostać potwierdzone, zapewniając dostępność systemu.

Powyższe własności stanowią fundament dla opracowywanych algorytmów konsensusu w sieciach blockchain. Jeżeli opracowany algorytm konsensusu nie spełniałby jednej z powyższych cech, to mówimy, że algorytm jest niepoprawny. Algorytmy konsensusu są znane od dłuższego czasu i szeroko zbadane na polu tradycyjnych systemów rozproszonych. Pierwsze sieci blockchain implementowały algorytmy takie jak proof-of-work (PoW) [24], proof-of-stake (PoS) [25] czy też problem bizantyjskich generałów (PBFT) [26]. Wraz z rozwojem sieci blockchain zostały zaprezentowane bardziej wysublimowane algorytmy takie jak DBFT [27], czy też algorytmy dla sieci opartych o skierowane grafy acykliczne (ang. DAG), np. Hashgraph [28].

Definicja 3.2.

Skierowany graf acykliczny to graf skierowany, który nie posiada cyklów skierowanych.

Definicja 3.3.

Graf skierowany G to para $G = \langle V, E \rangle$*, gdzie:*

1. $V \neq \emptyset$
2. $E \subseteq V \times V$

Algorytmy proof-of-work (PoW) oraz proof-of-stake (PoS) są oparte na koncepcji wyboru lidera – węzła sieci odpowiedzialnego za zatwierdzenie transakcji w danym momencie. W przypadku PoW wybór jest zależny od mocy obliczeniowej jaką dysponuje dany węzeł. Aby udowodnić posiadanie odpowiednich zasobów, węzły sieci rywalizują ze sobą rozwiązując skomplikowane zadanie odnalezienia zestawu danych, który pozwoli na wygenerowanie wartości skrótu SHA256 mniejszej lub równej od aktualnej wartości trudności wydobycia. Zadanie to jest wymagające w kontekście odnalezienia właściwego zestawu danych – wymaga wykonania obliczeń wartości funkcji skrótu dla wielkiej ilości możliwych danych wejściowych. Poprawność dokonanych obliczeń może być zweryfikowana przez uczestników sieci przy obliczeniu pojedynczego skrótu dostarczonych wartości. Węzeł sieci, który rozwiąże problem odnalezienia zestawu danych jako pierwszy zostaje wybrany jako lider, który zatwierdza transakcje dla danego bloku i pobiera wynagrodzenie w postaci wydobytych

środków i wniesionych przez użytkowników sieci opłat transakcyjnych. Wygenerowany blok, po prawidłowej weryfikacji, zostaje dodany do aktualnego stanu blockchain. Proces potwierdzania transakcji zakłada wykorzystanie najdłuższego łańcucha transakcji. W procesie kopania może dojść do powstania rozwidleń łańcucha, zatem jako właściwe wybierane jest to rozwidlenie, które ma największą długość. Blok jest uznawany za potwierdzony wtedy, gdy posiada co najmniej sześć bloków po nim następujących. Ten rodzaj konsensusu gwarantuje bardzo dobrą skalowalność i decentralizację. Cały proces opiera się na losowości doboru danych i jest identyczny dla wszystkich uczestników sieci. Za główny problem należy uznać prowadzony wyścig w zwiększaniu możliwości obliczeniowych węzłów sieci, który implikuje znaczące zwiększenie poboru energii elektrycznej i marnowania istniejących mocy obliczeniowych. Zgodnie z pracą [29] roczne zużycie energii elektrycznej przez sieć Bitcoin wynosi 127 TWh.

Algorytm konsensusu stosowany w Bitcoin cechował się niską przepustowością i wysokimi opóźnieniami. Powyższe skłoniło społeczność do opracowania jego następcy – Bitcoin-NG [30]. Bitcoin-NG wykorzystuje algorytm PoW stosowany uprzednio, nie zmieniając fazy wyboru lidera dokonującego zatwierdzania transakcji. Zmodyfikowany został zakres danych wykorzystywanych do obliczenia skrótu bloku – w przypadku Bitcoin-NG nie są brane pod uwagę dane transakcji. Algorytmy PoW są podatne na atak 51% sieci [24]. Jeżeli atakujący uzyska większość mocy obliczeniowych dla danej sieci wykorzystującej PoW, może zmodyfikować dane znajdujące się w blockchain i tym samym sfałszować zawartość rejestru.

W celu rozwiązania problemu ogromnej konsumpcji energii elektrycznej przez algorytmy PoW, zaproponowano algorytmy klasy PoS. Dla tych algorytmów, węzły rywalizują między sobą o prawo do zatwierdzania transakcji poprzez posiadane zasoby tokenów danej sieci. Koncepcja algorytmów PoS zakłada, że węzeł posiadający większe zasoby ma większe prawdopodobieństwo wybrania na lidera zatwierdzającego transakcje dla danego bloku. W najprostszej implementacji PoS można założyć, iż węzeł, który założył najwyższą stawkę zostaje liderem i to on zatwierdza dany blok. Można także dodać element PoW, w którym stawka ma wpływ na trudność znalezienia zestawu danych – im wyższa stawka i czas jej posiadania tym niższa trudność rozwiązania problemu PoW. Powyższe pozwala na znaczne ograniczenie zużycia zasobów, ale niesie ze sobą ryzyko powstania monopolu. Posiadacz największej ilości tokenów może trwale pełnić funkcję lidera, co prowadzi do centralizacji sieci. W pracy [31] zaproponowano modyfikację tego algorytmu określaną jako DPoS (Delegated Proof-of-Stake). Właściciel założonej stawki daje prawo innym węzłom sieci do

wybrania kandydatów na liderów dokonujących zatwierdzenia transakcji w danym bloku. Kandydaci z najwyższą liczbą głosów otrzymują prawo do zatwierdzenia bloku. Kolejne fazy – dodania bloku oraz potwierdzania transakcji są we wszystkich algorytmach PoS tożsame z algorytmami PoW.

W tym miejscu warto także wspomnieć o innych typach algorytmów konsensusu, które opierają się na wyborze lidera zatwierdzającego transakcje w danym bloku. Pierwszy z nich to proof-of-luck (PoL) [32], drugi to proof-of-burn (PoB) [33]. Proof-of-luck wykorzystuje zaufane środowisko uruchomieniowe (TEE) w celu wygenerowania wartości losowej do wyboru lidera, w przypadku proof-of-burn górnik niszczy część posiadanych przez siebie tokenów wysyłając je na adres, z którego nie jest możliwe ich rozdysponowanie. Górnik niszczący największą liczbę tokenów zostaje liderem.

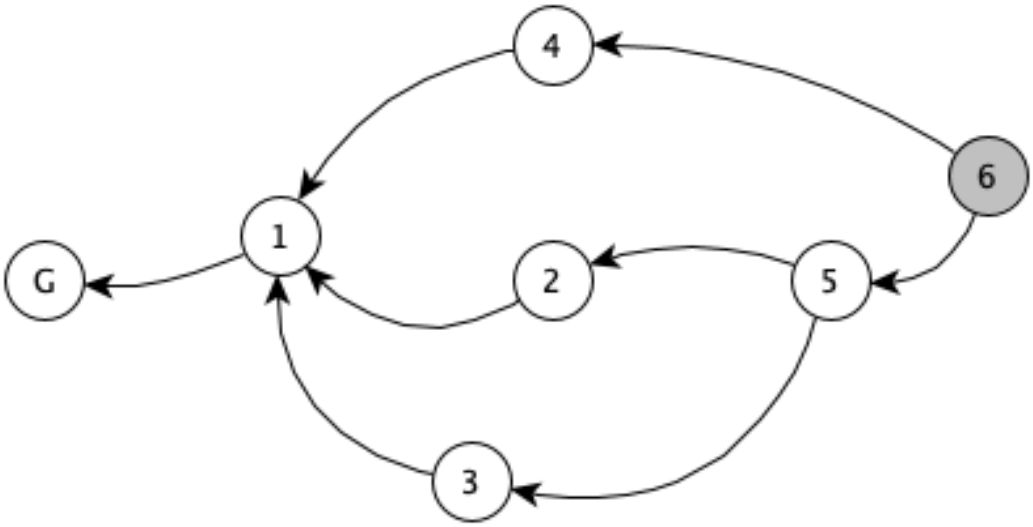
Tradycyjne systemy rozproszone zazwyczaj wykorzystują algorytmu konsensusu oparte o głosowanie. Algorytmy takie są bardzo intuicyjne – w momencie otrzymania zgody od większości głosujących, dane zostają zaakceptowane. W sieciach Blockchain, takich jak np. Hyperledger Fabric [34] stosowany jest algorytm PBFT. Algorytm ten został zaproponowany przez Castro i Liskov w 1999 [26]. Faza wyboru zatwierdzającego transakcje ma zazwyczaj charakter ankiety wypełnianej przez węzły. PBFT zakłada, że maksymalna liczba złośliwych węzłów sieci *f* nie przekracza 1/3 całkowitej liczby węzłów *n*, tj. $n \geq 3f + 1$.

Faza dodawania bloków w blockchain stosującym ten rodzaj konsensusu to trzyetapowy proces zatwierdzania:

- W pierwszym kroku, wybrany notariusz wysyła blok do innych węzłów sieci w celu jego weryfikacji.
- W drugim kroku, każdy z węzłów wysyła wynik weryfikacji do pozostałych węzłów sieci. Węzły potwierdzą zwrotnie poprawność bloku poprzez wysłanie odpowiedniej wiadomości. Jeżeli uzyskano więcej niż *2f+1* głosów „za" zatwierdzeniem, wtedy blok jest uznawany za ważny.
- W trzecim kroku, każdy z węzłów wysyła ponownie wynik weryfikacji do pozostałych węzłów sieci, a węzły te potwierdzają ostatecznie walidację bloku zgodnie z otrzymaną wiadomością. Jeżeli uzyskano więcej niż *2f+1* głosów „za" zatwierdzeniem, wtedy blok jest dodawany do blockchain.

Powyższe wskazuje na złożoność algorytmu PBFT na poziomie $O(n^2)$. W celu obniżenia złożoności obliczeniowej PBFT został zaproponowany algorytm SBFT [35]. Stanowi on

uproszczoną wersję algorytmu SBFT, w której faza dodawania bloku została podzielona na cztery etapy w stosunku do trzech dla PBFT. W kroku czwartym podpisywane są komunikaty kontrolne, a liczba *f+1* podpisanych komunikatów jest wystarczająca do zatwierdzenia danego bloku.



Rysunek 4 Splot dla IOTA [17]

Ostatni z omawianych w niniejszej rozprawie typów konsensusu jest oparty o zasadę „uczciwej rachunkowości” (ang. fair accounting). Jest on wykorzystywany w sieciach blockchain stosujących struktury skierowanych grafów acyklicznych (ang. DAG – Directed Acyclic Graph). Dla tego rodzaju Blockchain, faza wyboru węzła zatwierdzającego zakłada, że każdy z węzłów sieci jest zatwierdzającym swoje własne transakcje. Oznacza to, że każdy z węzłów sieci ma podobne szanse na zostanie zatwierdzającym. W fazie dodawania bloku zostają zdefiniowane zasady dotyczące powiązania bloków z ich blokami nadrzędnymi. Blok zawsze ma więcej niż jeden blok nadrzędny, dlatego też ta struktura tworzy skierowany graf acykliczny, nie łańcuch. W fazie zatwierdzania transakcji wykorzystywane są bardziej złożone reguły, które mają za zadanie określić, czy blok osiągnął konsensus. W tym celu pod uwagę brane jest m.in. położenie bloku w strukturze grafu.

Rysunek 4 przedstawia strukturę splotu wykorzystywaną w blockchain IOTA. Każda transakcja w splocie jest reprezentowana przez wierzchołek. Wierzchołek, który nie posiada poprzedników jest nazywany szczytem. Na rysunku 4 szczytem jest wierzchołek o numerze 6.

W fazie dodania bloku nowy wierzchołek jest dodawany do splotu i musi wybrać dwa istniejące szczyty jako swoich następców. Dla IOTA można wyróżnić dwa typy zatwierdzenia transakcji:

- całkowity,
- częściowy.

Transakcja, która została pośrednio lub bezpośrednio potwierdzona przez wszystkie szczyty jest potwierdzona całkowicie. Jeżeli nie jest to możliwe, np. ze względu na powstałe w sieci opóźnienia, wtedy wymagane jest, aby pewna część, np. 80% szczytów dokonało potwierdzenia, które będzie potwierdzeniem częściowym.

4. Rozwój technologii opartych o rejestr rozproszony

W tym rozdziale przybliżone zostały kluczowe ewolucje i zastosowania technologii blockchain, w szczególności generacje 2.0 i 3.0. W pierwszym podrozdziale omówiony jest projekt Ethereum, będący reprezentantem drugiej generacji blockchain. Ethereum implementuje smart kontrakty, które zrewolucjonizowały możliwości zastosowania rejestrów rozproszonych. W następnym podrozdziale została przedstawiona specyfikacja blockchain IOTA, przedstawiciela trzeciej generacji blockchain dedykowanego zastosowaniom związanym z Internetem Rzeczy (ang. Internet of Things, IoT). W kolejnym podrozdziale została przedstawiona specyfikacja Hyperledger Fabric – najbardziej rozpowszechnionej platformy do tworzenia prywatnych blockchainów, znajdujących zastosowanie w systemach korporacyjnych. Rozdział przedstawia także technologie zdecentralizowanego przechowywania danych, które stanowią alternatywne rozwiązanie dla bezpiecznego i rozproszonego przechowywania danych. W ostatnim podrozdziale przedstawione zostały możliwości praktycznego zastosowania technologii blockchain 2.0 i 3.0, pokazując jej wpływ na różne sektory gospodarki.

4.1. Blockchain 2.0 – Ethereum

Niniejszy podrozdział zawiera opis struktury i działania sieci Ethereum, w tym mechanizmy przejścia stanów oraz różnice w typach kont. Wyjaśniono sposób obsługi transakcji i wiadomości w sieci, wskazując zarówno proces ich wykonania jak i walidacji w kontekście wydobywania nowych bloków. Przybliżona zastała specyfika smart kontraktów oraz ich wpływa na interakcje w sieci.

Koncepcja Ethereum opisana w pracach [36][37] ma swoje początki w roku 2013, kiedy to Vitalik Buterin oraz Gary Wood zaproponowali koncepcję nowej generacji blockchain. Podstawowym założeniem było wprowadzenie własnego języka programowania, który jest językiem kompletnym w sensie Turinga[15]. Język ten miał pozwolić na implementację kontraktów, które mogą być wykorzystane do opisu funkcji przejścia stanu. Powyższe pozwala na tworzenie nowych zastosowań dla blockchain, co znacząco zwiększa możliwości jego wykorzystania. Tworzone w ten sposób rozproszone aplikacje rozszerzają

[15] Językiem kompletnym w sensie Turinga nazywany jest język programowania, który jest w stanie symulować dowolną maszynę Turinga, a tym samym może wykonać wszystkie możliwe obliczenia.

funkcjonalność blockchain, dając programistom dowolność w kształtowaniu zasad własności, formatów transakcji czy funkcji zmiany stanu.

Stan sieci Ethereum jest definiowany poprzez konta. Każde konto posiada dwudziestobajtowy adres oraz przejścia stanu będące bezpośrednimi transferami wartości i informacji pomiędzy kontami. Konto w sieci Ethereum składa się z następujących czterech pól:

- *Nonce* – licznik wykorzystywany by zapewnić, że każda transakcja jest przetwarzana tylko jeden raz.
- Saldo konta wyrażone w ETH.
- Kod kontraktu danego konta, o ile występuje.
- Przestrzeń danych konta, domyślnie pusta.

Wyróżnia się dwa główne rodzaje kont w sieci Ethereum – konta kontraktów zarządzane przez kod kontraktu oraz konta użytkowników zewnętrznych, które są kontrolowane przy wykorzystaniu posiadanych przez nich kluczy prywatnych. Konta użytkowników zewnętrznych nie mają żadnego kodu. Pozwalają na wysyłanie wiadomości, które wymagają utworzenia i podpisania transakcji. W przypadku konta kontraktu, kontrakt jest aktywowany każdorazowo po otrzymaniu wiadomości, pozwalając na odczyt i zapis oraz odczyt wewnętrznej przestrzeni danych, a także wysyłanie wiadomości lub tworzenie innych kontraktów w rezultacie powyższego. Kontrakt w sieci Ethereum nie stanowi logiki, która nakłada pewne ograniczenia na korzystanie z sieci i musi być spełniona. Kontrakt powinien być utożsamiany z autonomicznym użytkownikiem, który jest całkowicie osadzony w środowisku wykonawczym sieci Ethereum i zawsze wykonuje zadane czynności po wzbudzeniu przez wiadomość lub transakcję. Kontrakt posiada pełną kontrolę nad swoim saldem ETH oraz przestrzenią danych wykorzystywaną do przechowywania wartości jego zmiennych.

Sieć Ethereum określa transakcję jako podpisaną paczkę danych przechowującą wiadomość przesyłaną przez konto użytkownika zewnętrznego. Transakcja zawiera:

- Dane odbiorcy wiadomości.
- Podpis identyfikujący jej nadawcę.
- Ilość ETH transferowanych pomiędzy nadawcą i odbiorcą.
- Opcjonalne pole danych.
- Wartość *STARTGAS*, reprezentującą maksymalną liczbę kroków obliczeniowych jakie mogą być wykonana podczas wykonania transakcji.

- Wartość *GASPRICE*, określającą wysokość opłaty jaką nadawca jest skłonny uiścić za każdy krok obliczeniowy.

Pierwsze trzy pola występują w transakcjach większości blockchain implementujących kryptowaluty. Pole danych jest domyślnie puste, jednakże może zawierać dane dostępne dla kontraktu. Przykładem może być tu kontrakt implementujący rejestrację nazw domenowych. Taki kontrakt może zawierać dane, które są interpretowane jako pola do rejestracji nazwy domenowej – pierwsze może odpowiadać nazwie domenowej, drugie powiązanemu z nią adresowi IP. Taki kontrakt będzie pobierał dane przesłane w odpowiedniej wiadomości i zapisywał je w przestrzeni danych konta. Pola *STARTGAS* i *GASPRICE* są wykorzystywane przez sieć Ethereum do zapewnienia dostępności usług. Określenie maksymalnej ilości kroków pozwala na wyeliminowanie wykonywania zapętlonego kodu lub też nadużywania zasobów obliczeniowych sieci poprzez wykorzystanie kodu nieefektywnego. Za podstawową jednostkę rozliczeniową jest przyjęty *gaz*[16]. Jeden krok obliczeniowy to zazwyczaj koszt jednej jednostki gazu. W przypadku bardziej skomplikowanych operacji koszt jednego kroku może wynosić kilka jednostek lub więcej. Dodatkowo, sieć pobiera opłatę w wysokości pięciu jednostek gazu za każdy przechowywany bajt danych transakcji. Celem powyższego jest redukcja wykorzystania zasobów sieci przez nieuczciwych użytkowników. Wraz ze wzrostem wykorzystania sieci, muszą oni ponosić wyższe opłaty za przechowanie danych, co pozwala na znaczną redukcję potencjalnych nadużyć.

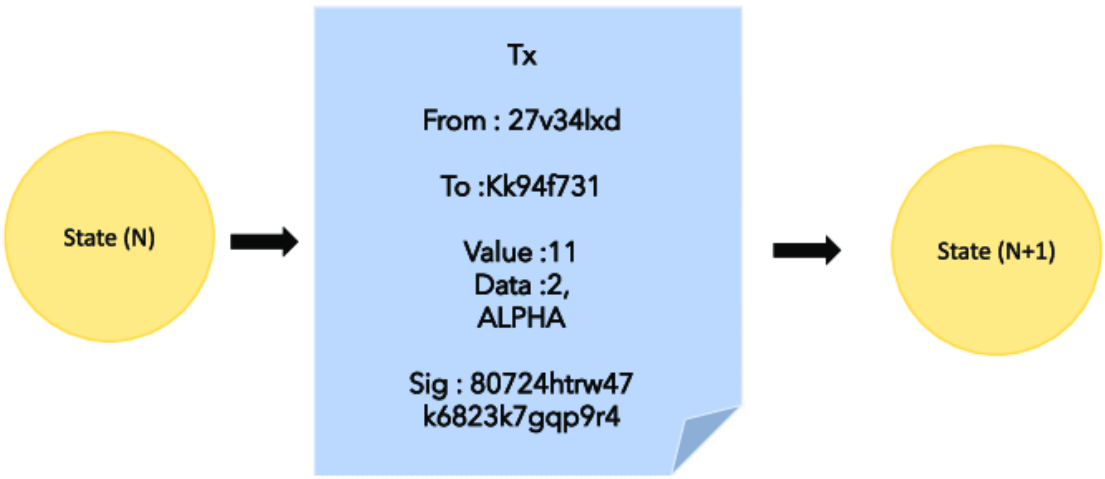
Sieć Ethereum pozwala kontraktom na przesyłanie wiadomości do innych kontraktów sieci. Wiadomości są wirtualnymi obiektami, które nie podlegają serializacji[17] i istnieją jedynie wewnątrz środowiska wykonawczego Ethereum. Każda wiadomość składa się z następujących pól:

- Identyfikatora nadawcy wiadomości,
- Identyfikatora odbiorcy wiadomości,
- Ilości ETH transferowanych wraz z wiadomością,
- Pole danych (opcjonalne),
- Wartości *STARTGAS.*

[16] Gaz w sieci Ethereum stanowi jednostkę miary, która jest wykorzystywana do kwantyfikacji ilości zasobów obliczeniowych i pamięciowych wymaganych do wykonania operacji, takich jak transakcje i wykonanie smart kontraktów.

[17] Serializacja to proces konwersji struktury danych lub obiektu stanu do formatu, który może być zapisany (na przykład w pliku lub buforze pamięci) lub przesłany (na przykład przez sieć transmisji danych).

Wiadomość może być interpretowana podobnie jak transakcja, z zastrzeżeniem metody jej generowania. Transakcje są tworzone przez zewnętrznych użytkowników sieci, podczas gdy wiadomości mogą być tworzone jedynie przez kontrakty. Kontrakt generuje wiadomość poprzez wykonanie kodu operacji *CALL*, którego celem jest wygenerowanie oraz wykonanie wiadomości. Wiadomości dają możliwość uzyskania takiej samej relacji pomiędzy kontraktami, jak pomiędzy zewnętrznymi użytkownikami sieci. Pozwala to na pełną autonomię kontraktów i realizację przez nie zaawansowanej logiki. Określony limit gazu obejmuje daną transakcję oraz wszystkie jej kolejne podwykonania. Jeżeli użytkownik A wysyła transakcję do użytkownika B, określając ilość gazu na 1000 jednostek, a B zużyje 600 jednostek zanim przekaże wiadomość do C, przy założeniu, że wewnętrzne zużycia gazu przez wykonanie C wyniesie 300, wtedy B będzie mógł zużyć jeszcze 100 jednostek gazu zanim limit zostanie wyczerpany.



Rysunek 5 Funkcja przejścia stanu Ethereum [38]

Funkcja przejścia stanu dla Ethereum, określana jako

$$APPLY(State[N], Tx) \rightarrow State[N+1]$$

jest definiowana następująco:

1. Weryfikacja poprawności transakcji – walidacja ilości pól, poprawności podpisu, porównanie wartości nonce z nonce konta nadawcy. Jeżeli walidacja będzie negatywna, funkcja zwraca błąd.
2. Obliczanie wartości opłaty transakcyjnej z wykorzystaniem formuły $FEE = STARTGAS * GASPRICE$. Określenie adresu nadawcy na podstawie

podpisu. Opłata jest odejmowana od salda konta nadawcy, a wartość jego nonce jest inkrementowana. Jeżeli nadawca nie posiada odpowiedniej ilości środków, funkcja zwraca błąd.

3. Przypisz $GAS = STARTGAS$ i pobierz odpowiednią wartość gazu za każdy bajt danych zawartych w transakcji.
4. Przetransferuj środki w wysokości określonej w transakcji pomiędzy kontem nadawcy a kontem odbiorcy. Jeżeli konto odbiorcy nie istnieje, utwórz nowe. Jeżeli konto odbiorcy to kontrakt, uruchom kod kontraktu aż do poprawnego zakończenia lub wyczerpania gazu.
5. Jeżeli nadawca nie posiadał wystarczającej liczby środków, albo wykonanie kodu wyczerpało dostępny gaz, to wszystkie zmiany wprowadzane przez transakcje powinny zostać wycofane. Jedynie pobranie opłaty za wykonanie oraz dodanie tej opłaty do nagrody za wydobycie bloku jest pozostawiane w nowym stanie.
6. W przeciwnym przypadku zwróć poniesione opłaty oraz niewykorzystany gaz do nadawcy. Opłaty za wykorzystany gaz przekaż w nagrodzie za wydobycie bloku.

Przyjmijmy następujący kod, dla którego dokonana zostanie analiza funkcji przejścia stanu:

```
if !self.storage[calldataload(0)]:
    self.storage[calldataload(0)] = calldataload(32)
```

Powyższy kod nie jest bezpośrednio wykonywany w EVM (Ethereum Virtual Machine) [39], lecz jest kodem napisany w języku Serpent [40], obok Solidity [41] – jednym z języków wysokopoziomowych wykorzystywanych do tworzenia kodu kontraktów Ethereum. Rzeczywiście wykonywany kod ma charakter kodu niskopoziomowego, interpretowanego przez EVM. Przyjęto założenie, że przestrzeń danych kontraktu jest pusta, transakcja przesyła 10 ETH, limit gazu to 2000, cena jednostki gazu to 0.001 ETH, a przesyłane dane składają się z 32 bajtów odpowiadających liczbie 2 i kolejnych 32 bajtów tworzących ciąg tekstowy CHARLIE. W tym przypadku funkcja przejścia stanu będzie wykonywała następujące czynności:

1. Przeprowadzenie walidacji danych zawartych w transakcji.

2. Zweryfikowanie, że nadawca transakcji posiada saldo co najmniej $2000 \cdot 0.001 = 2\,ETH$. Jeżeli weryfikacja przebiegła pozytywnie, od salda nadawcy odejmowane jest 2 ETH.
3. Zakładając, iż początkowe saldo gazu wynosiło 2000, transakcja ma 170 bajtów a opłata za każdy bajt to 5, pozostałe saldo gazu wyniesie:
$$2000 - 5 \cdot 170 = 1150.$$
4. Pomniejszenie salda konta nadawcy o 10 ETH i dodanie tej wartości do salda odbiorcy.
5. Wykonanie kodu kontraktu. W niniejszym przykładzie jest to pobranie wartości 2 znajdujące się na pierwszych 32 bajtach danych transakcji i wykorzystanie tej wartości jako indeks w przestrzeni danych kontraktu. Dla analizowanego kontraktu wartość te nie była ustawiona, zatem przypisywana jest do indeksu 2 wartość CHARLIE znajdująca się na pozostałych 32 bajtach danych transakcji. Przyjmując, że wykonanie tego kodu wymagało zużycia 187 jednostek gazu, pozostałe jego saldo wyniesie $1150 - 187 = 963$.
6. Zwrócenie niewykorzystanego gazu w wysokości
$$963 \cdot 0{,}001\,ETH = 0{,}963\,ETH$$
na konto nadawcy oraz zwrócenie aktualnego stanu sieci.

Jeżeli powyższa transakcja zostałaby wysłana na konto nienależące do kontraktu, wtedy całkowita opłata transakcyjna byłaby równa iloczynowi liczby bitów danych transakcji oraz określonej ceny gazu. Należy zaznaczyć, iż sposób wycofywania wiadomości jest taki sam jak transakcji. Jeżeli wykonanie wiadomości wyczerpie przeznaczony na ten cel limit gazu, jej wykonanie, jak i wykonanie pochodnych wiadomości zostanie wycofane, pozostawiając jedynie główną wiadomość. Oznacza to, że kontrakt może wywołać inny kontrakt. Jeżeli A wywoła B, korzystając z G gazu, wtedy wykonanie A zużyje nie więcej niż G gazu.

Kod kontraktów Ethereum jest pisany z wykorzystaniem niskopoziomowego języka opartego o kod bajtowy [42]. Kod ten składa się z serii bajtów, które reprezentują operacje. Wykonanie kodu to nieskończona pętla, która składa się z cyklicznie powtarzanych operacji dla zadanego licznika programu i jego inkrementacji o jeden aż do zakończenia kodu, wystąpienia błędu lub napotkania kodów *STOP* lub *RETURN*. Wykonywane przez kod operacje mają dostęp do trzech przestrzeni, w których mogą być składowane dane:

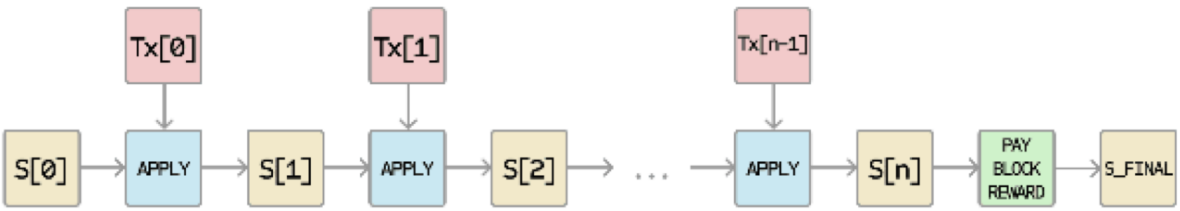
- Stosu, czyli kontenera zbudowanego w oparciu o regułę LIFO (Last In First Out). Wartości mogą być odkładane lub zdejmowane ze stosu.

- Pamięci, która stanowi nieskończoną tablicę bajtów.
- Przestrzeni kontraktu przeznaczonej na przechowywanie wartości w postaci par klucz/wartość. W odróżnieniu od pozostałych rodzajów przestrzeni, ta zachowuje wartości także po zakończeniu wykonania kodu kontraktu.

Kod posiada dostęp do danych przychodzącej wiadomości, takich jak wartość transferu, identyfikator nadawcy, załączone dane, a także do danych znajdujących się w nagłówku danego bloku. Kontrakt może zwracać dane w postaci tablicy bajtów.

Formalny opis modelu wykonania kodu EVM może zostać sprowadzony do poniższych punktów:

- Stan pracującej EVM może zostać wyrażony za pomocą następującej krotki `(block_state, transaction, message, code, memory, stack, pc, gas)`.
- `block_state` to globalny stan sieci zawierający wszystkie konta, ich salda oraz przechowywane dane.
- Każda kolejna runda wykonania kodu rozpoczyna się od odnalezienia instrukcji wskazywanej przez bajt nr `pc` wartości `code`. Jeżeli wartość `pc` jest większa lub równa długości kodu, wtedy wybierany jest bajt 0. Każda instrukcja wpływa na stan krotki w zdefiniowany dla niej sposób. Przykładowo, instrukcja *ADD* zdejmuje dwa elementy ze stosu i odkłada ich sumę, redukując przy tym gaz o 1 i inkrementując `pc` o 1. Dla instrukcji *SSTORE* zdejmowane są dwa elementy ze stosu, a drugi z tych elementów jest umieszczony w przestrzeni danych kontraktu pod indeksem wskazywanym przez pierwszy element.



Rysunek 6 Proces wydobywania bloku Ethereum [36]

Struktura blockchain Ethereum jest podobna do struktury Bitcoin, jednakże w przypadku Ethereum bloki zawierają kopię zarówno listy transakcji, jak i ostatniego stanu sieci. Każdy blok zawiera także informację o swoim numerze oraz trudności wydobycia. Podstawowy

algorytm walidacji bloku Ethereum przy wykorzystaniu algorytmu konsensusu PoW przebiega następująco:

1. Weryfikacja istnienia poprzedzającego bloku oraz jego poprawności.
2. Weryfikacja znacznika czasowego bloku. Jego wartość musi być większa niż bloku poprzedzającego i nie większa o 15 minut od tej wartości.
3. Weryfikacja numeru bloku, trudności, korzenia transakcji oraz limitu gazu pod kątem poprawności.
4. Weryfikacja poprawności rozwiązania PoW dla bloku.
5. Założenie `S[0]` jako stan sieci na koniec poprzedniego bloku.
6. Założenie, że `TX` stanowi listę transakcji bloku posiadająca `n` transakcji. Dla każdej liczby całkowitej i w zakresie $0, \dots, n-1$ ustaw `S[i+1]=APPLY(S[i],TX[i])`. Jeżeli aplikacja zwróci błąd albo jeżeli cały limit gazu zostanie wykorzystany, zwróć błąd.
7. Niech `S_FINAL` będzie równe `S[n]` z dodaną nagrodą za wydobycie bloku.
8. Weryfikacja czy korzeń drzewa Merkle dla stanu `S_FINAL` jest równy korzeniowi dla końcowego stanu określonemu w nagłówku bloku. Jeżeli równość jest spełniona, blok jest ważny. W przeciwnym przypadku blok jest nieważny.

Powyższy proces zakłada przechowywanie całego stanu sieci w każdym z bloków w strukturze drzewiastej. Po wydobyciu nowego bloku do drzewa jest dodawana jedynie niewielka ilość informacji dotyczących nowych modyfikacji stanu, co pozwala na zmniejszenie liczby wprowadzanych zmian i zwiększenie efektywności. Wynika to z faktu przechowywania niemalże tych samych informacji przez dwa przylegające do siebie bloki, zatem dane które już raz zostały zapisane mogą być powiązane przez wskaźniki. Wykorzystane tu rozwiązanie to drzewo Patricia [43], które zakłada modyfikację drzewa Merkle[18] poprzez dopuszczenie wstawiania i usuwania węzłów do drzewa, a nie tylko ich modyfikacji. Większość informacji jest dostępna w ostatnim bloku, dlatego nie ma potrzeby przechowywania całej historii blockchain.

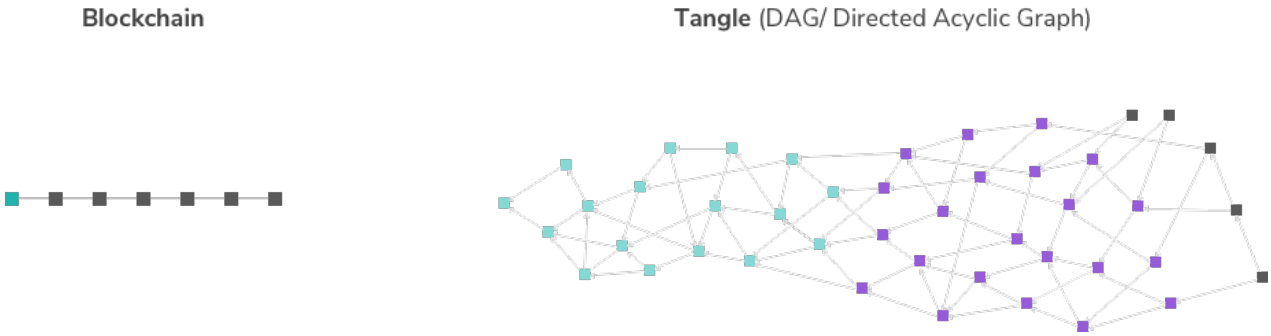
[18] Drzewo Merkle, znane również jako drzewo skrótów, to struktura danych stosowana w kryptografii, składająca się z zestawu węzłów, gdzie każdemu węzłowi będącemu liściem przypisany jest skrót kryptograficzny danego bloku danych, a każdy węzeł niebędący liściem zawiera skrót kryptograficzny wynikający z połączenia skrótów kryptograficznych jego dzieci, co umożliwia efektywne i bezpieczne weryfikowanie zawartości dużych zbiorów danych.

Kod smart kontraktów jest wykonywany w funkcji przejścia stanu. Wykonanie kodu jest częścią procesu walidacji bloków, zatem aby dodać transakcję do pewnego bloku B, kod ten musi zostać wykonany przez wszystkie węzły sieci. Tylko takie wykonanie pozwala na zwalidowanie bloku i zapisanie go w lokalnej historii sieci.

4.2. Blockchain 3.0 – IOTA

Podrozdział ten omawia **kryptowalutę IOTA** przeznaczoną specjalnie do zastosowania dla Internetu rzeczy. IOTA wyróżnia się zastosowaniem odmiennej struktury danych dla rejestru rozproszonego. Jest to skierowany graf acykliczny, nazywany przez autorów IOTA splotem. Przedstawiona została specyfikacja najważniejszych algorytmów składowych, takich jak algorytm wyboru szczytu (Tip Selection Algorithm, TSA) oraz protokół zapewniający bezpieczne przesyłanie danych w sieci publicznej – Masked Authenticated Messaging (MAM).

IOTA [44] to kryptowaluta pozwalająca na wysoką skalowalność i niskokosztowe transfery środków pomiędzy użytkownikami sieci. Jej podstawowym przeznaczeniem jest Internet rzeczy (Internet of Things, IoT). W odróżnieniu od sieci takich jak Bitcoin czy Ethereum wykorzystujących strukturę łańcucha bloków, IOTA stosuje skierowany graf acykliczny (Directed Acyclic Graph), nazywany przez twórców splotem (tangle). Transakcje reprezentowane są przez wierzchołki splotu, nazywane stronami (site). Istnienie bezpośredniej krawędzi od jednej strony do następnej oznacza, że strona źródłowa zaakceptowała stronę docelową. IOTA nie stosuje fazy wydobywania bloków znanej z sieci takich jak Bitcoin. Konsensus jest osiągany na każdym z węzłów poprzez aktualizację lokalnego rejestru. Aktualizacja jest dokonywana natychmiast po dołączeniu nowej strony do splotu i jej akceptacji przez istniejące strony rejestru. Strona, która dokonuje akceptacji nazywana jest szczytem (tip).



Rysunek 7 Blockchain (Bitcoin, Ethereum) vs Splot IOTA [48]

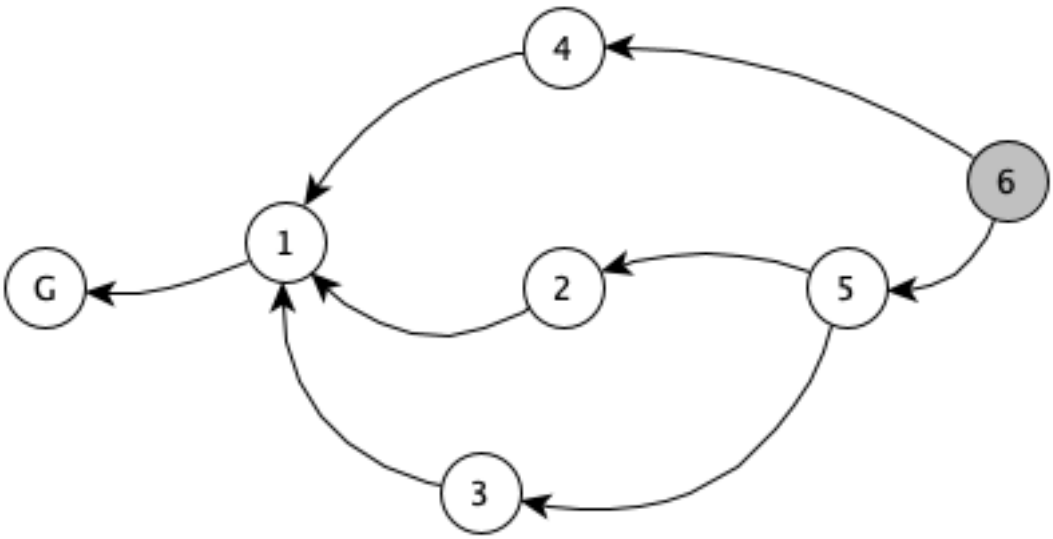
Wybór szczytów jest dokonywany z wykorzystaniem algorytmu wyboru szczytu (TSA – Tip Selection Algorithm) [45]. Algorytm ten wykonuje dwa losowe ważone przejścia[19] przez splot, identyfikując dwa wybrane szczyty. IOTA zakłada otrzymywanie transakcji przez

[19] Ważone przejście w kontekście algorytmu wyboru szczytu (TSA) w IOTA odnosi się do procesu, w którym wybór transakcji do zatwierdzenia jest dokonywany na podstawie losowego przeszukiwania grafu splotu, przy czym prawdopodobieństwo wyboru konkretnej transakcji jest proporcjonalne do akumulowanej wagi, będącej miarą liczby bezpośrednich i pośrednich zatwierdzeń danej transakcji przez inne transakcje w sieci.

wszystkie węzły sieci, które następnie są odpowiedzialne za ich dodanie do splotu i dalsze dystrybuowanie do sąsiednich wierzchołków. Ponieważ użytkownicy sieci samodzielnie weryfikują transakcję, sieć nie wymaga istnienia górników [49]. Wszyscy uczestnicy sieci pełnią taką samą rolę. Mogą tworzyć i walidować transakcję i mają taki sam wpływ na ustalenie konsensusu sieci. Założenie takie pozwala na znaczące zmniejszenie kosztu walidacji transakcji, ponieważ wymaga jedynie akceptacji dwóch innych transakcji w sieci.

IOTA podobnie jak inne rejestry rozproszone musiała rozwiązać problem podwójnego wydawania środków [50]. Splot sieci IOTA składa się z węzłów, które mogą tworzyć i walidować transakcje. Każdy z węzłów reprezentuje transakcję, patrz [46]. Proces dodania transakcji do splotu zakłada wykonanie poniższych kroków:

1. Węzeł wybiera dwie transakcje do zatwierdzenia. Wybór jest dokonywany z wykorzystaniem algorytmu TSA.
2. Węzeł weryfikuje czy wybrane transakcje zawierają transakcje podwójnego wydania środków. Jeżeli tak, transakcje są odrzucane.
3. Węzeł rozwiązuje algorytm PoW poprzez odnalezienie *nonce* generującego oczekiwany skrót kryptograficzny z danymi zatwierdzanej transakcji.
4. Użytkownik wysyła transakcję do sieci. Transakcja ma status niezatwierdzonej i stanowi szczyt (tip) splotu.
5. Szczyt czeka na zatwierdzenie. Może ono nastąpić w sposób bezpośredni lub pośredni.



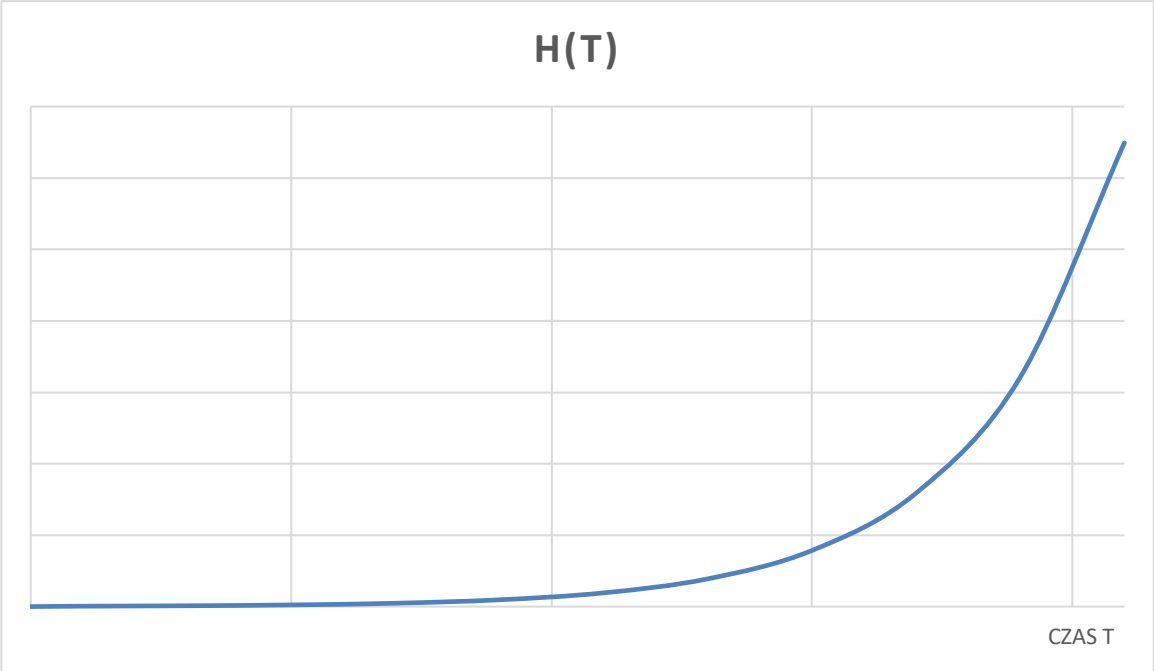
Rysunek 8 Splot IOTA [17]

Transakcje pojawiają się w sposób asynchroniczny, co powoduje, że węzły nie mają dostępu do wszystkich transakcji w sieci. W przypadku rysunku 8, węzeł 4 nie zna historii dla transakcji 2, 3 oraz 5. Dla powyższego stanu mogą istnieć potencjalne transakcje podwójnego wydawania środków – np. 4 oraz 2. Te transakcje nie były poddane wspólnej walidacji. Zgodnie z pracami [44, 51, 52] wybór pomiędzy skonfliktowanymi transakcjami jest dokonywany poprzez wykonanie algorytmu TSA wielokrotnie. W każdym wykonaniu określa się, która z transakcji ma większe prawdopodobieństwo pośredniej akceptacji przez wybrany szczyt splotu. Gałąź o wyższym prawdopodobieństwie jest uznawana za prawidłową, a ta o prawdopodobieństwie niższym jest odrzucana.

Algorytm konsensusu dla splotu jest ściśle powiązany z akumulowaną wagą. Waga jest proporcjonalna do nakładu pracy poświęconego na walidację danej transakcji. Implementacja [44] określa jej wartość na proporcjonalną do 3^n, gdzie *n* jest liczbą naturalną. Transakcja o wyższej wadze jest transakcją o wyższej ważności dla sieci. Akumulowana waga jest zdefiniowana jako suma wagi danej transakcji oraz wszystkich transakcji, które w sposób bezpośredni lub pośredni ją zatwierdzają. Oznacza to, iż gałęzie o większej wadze będą miały tendencję do dalszego rozrastania. Gałęzie o mniejszej wadze będą izolowane i prawdopodobieństwo ich wyboru będzie niskie. Konsekwencją tych założeń jest, że to szczyty będą najczęściej dokonywać walidacji transakcji.

Osiągnięcie konsensusu jest zależne od prawdopodobieństwa tego, że dana transakcja nie może zostać odrzucona. W zależności od poziomu istotności transakcji jej akceptacja jest możliwa, jeżeli jej gałąź zostanie wybrana prze algorytm wyboru szczytów wystarczająco wiele razy. Akumulowana dla transakcji powinna rosnąć w sposób proporcjonalny do liczby nowo dołączanych szczytów. Każdy ze szczytów jest powiązany z tą transakcją w sposób bezpośredni lub pośredni, co skutkuje zwiększeniem jej wagi. Poniższe równanie [44] przedstawia akumulowaną wagę *H(t)* transakcji jako funkcję czasu *h*. *h* jest średnim czasem jaki jest wymagany do przeprowadzenia obliczeń przez urządzenia do wygenerowania transakcji, *t* oznacza czas.

$$H(t) = 2e^{\frac{0{,}352t}{h}}$$



Rysunek 9 Zależność akumulowanej wagi od czasu

Na rysunku 9 można zauważyć, iż wraz ze wzrostem wartości czasu t wzrasta wartość akumulowanej wagi. Początkowo jest to przyrost prawie liniowy i okres ten jest nazywany okresem adaptacji. Jeżeli transakcja nie zostanie odrzucona w tym okresie, wartość akumulowanej wagi zaczyna gwałtownie rosnąć, co wynika z dołączania do niej kolejnych szczytów splotu, wybierających tą transakcję i ją potwierdzających ze względu na wysokie prawdopodobieństwo pozostania jej w sieci. Ponieważ transakcja zawsze jest umieszczana w sieci, to usługi mogą zdecydować się na akceptowanie stanu splotu z tą transakcją. Usługa może określić poziom ufności od którego dana transakcja będzie akceptowana. W przypadku mikrotransakcji jest to zazwyczaj poziom powyżej 50%, natomiast dla giełd i kantorów wartość ta wynosi powyżej 99%.

Algorytm TSA wykorzystuje próbkowanie Monte Carlo łańcuchami Markowa[20] [53], które w dalszej części tej pracy będzie określane jako MCMC. Wykorzystanie tej metody do wyboru szczytów bazuje na założeniu, że główny łańcuch splotu ma większą wagę niż potencjalni atakujący. Dodatkowo w [44] określono, że waga nie może być jedynym parametrem, ale powinien być także wykorzystany czynnik α, który nie pozwoli na ograniczenie wyboru do szczytu o największej wadze.

[20] Próbkowanie Monte Carlo łańcuchami Markowa (MCMC) to technika statystyczna używana do estymacji rozkładów prawdopodobieństwa poprzez generowanie serii próbek z sekwencyjnie powiązanych rozkładów, gdzie każda próbka jest zależna od poprzedniej.

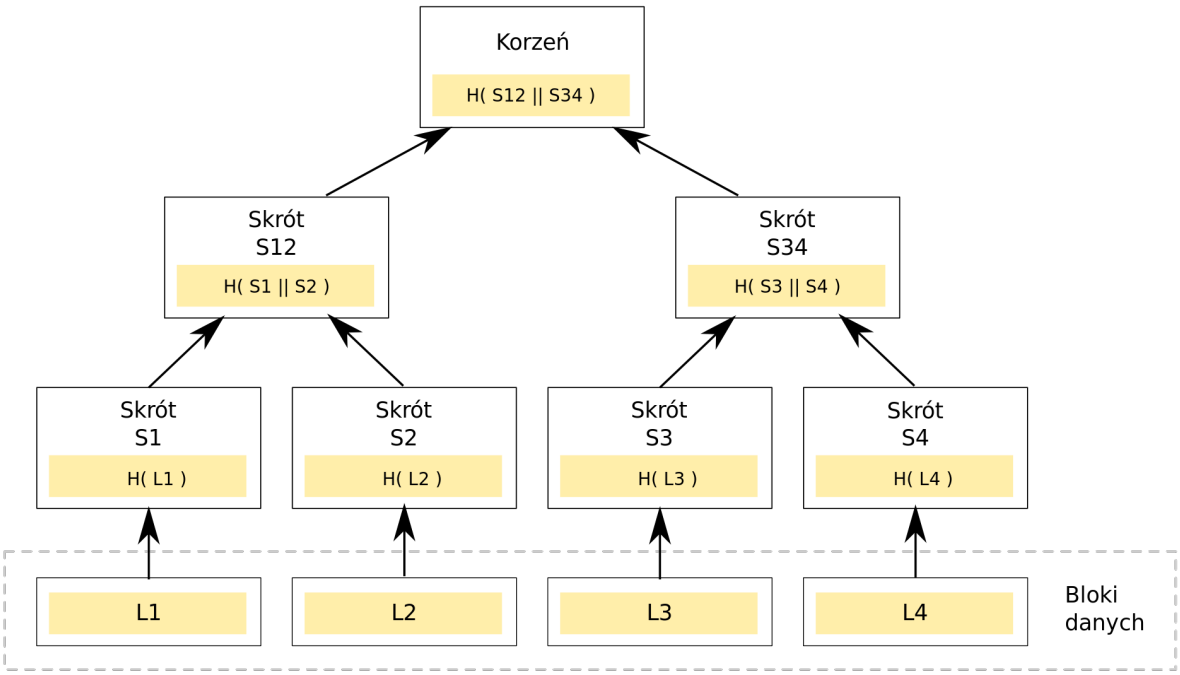
Praktyczna implementacja splotu stawia wiele wyzwań. Do najważniejszych należy zaliczyć rozmiar węzłów sieci, liczbę pośrednich potwierdzeń transakcji oraz zagadnienia związane z prywatnością danych przetwarzanych w transakcjach. IOTA rozwiązuje problemy skalowalności sieci poprzez wprowadzenie roli koordynatora. Koordynator ma za zadanie zabezpieczyć sieć przed nieuczciwymi uczestnikami, którzy chcieliby ją zaatakować. Algorytm konsensusu IOTA przewiduje, że każda potwierdzona transakcja powinna mieć pośrednią lub bezpośrednią referencję do transakcji podpisanej przez koordynatora [54]. Taka transakcja jest emitowana co dwie minuty, a każda transakcja podpisana przez nią jest traktowana jako potwierdzona w 100% [52]. Rolą koordynatora jest także weryfikowanie, czy w sieci nie nastąpił proceder podwójnego wydawania środków. Rola koordynatora wprowadza do sieci centralizację, która zwiększa bezpieczeństwo i skalowalność sieci, ale jednocześnie w negatywny sposób rzutuje na odczucia użytkowników. Koordynator posiada bowiem możliwość nadania transakcjom priorytetu, może zamrozić wybrane środki czy też zignorować wybrane transakcje. Koordynator stanowi także centralny punkt, który jest narażony na ataki. W przypadku, gdyby koordynator przestał działać, wszystkie potwierdzenia w sieci zostaną wstrzymane.

Sieć IOTA pozwala na przesyłanie transakcji, w których nie są transferowane żadne środki, a jedynie dane. Duża ilość wymienianych danych powoduje, że sieć rozrasta się do rozmiarów utrudniających jej przetwarzanie na węzłach. Problem ten jest rozwiązywany przez tworzenie migawek stanu sieci, podczas których zachowywane są jedynie salda adresów, a wszystkie pozostałe dane są usuwane. Migawki są wykonywane przez węzły sieci, w sposób ręczny lub całkowicie zautomatyzowany.

Definicja 4.1.

Migawka (snapshot) w kontekście sieci IOTA to proces zapisywania aktualnego stanu sald wszystkich adresów, przy jednoczesnym usuwaniu wszystkich innych danych transakcyjnych z sieci, co pozwala na redukcję objętości danych i usprawnienie przetwarzania przez węzły, zarówno w trybie ręcznym, jak i zautomatyzowanym.

W przypadku niektórych aplikacji istnieje konieczność przechowywania całej historii transakcji, np. w celu wprowadzenia audytowalności i przejrzystości działań. Powyższe jest realizowane z wykorzystaniem tzw. permanode (permanentnych węzłów) [55], które przechowują kompletną historię danych w splocie, uniemożliwiając ich modyfikację lub usunięcie.



Rysunek 10 Drzewo Merkle [57]

IOTA została zaprojektowana z myślą o wykorzystaniu do urządzeń pracujących w IoT. Powyższe znajduje swoje odzwierciedlenie w możliwości wymiany danych w transakcjach, w których nie zachodzi transfer środków. Adres może zostać wykorzystany do przechowywania danych przesyłanych przez urządzenia. Transakcje w sieci są publiczne, zatem potencjalny atakujący mógłby dokonać modyfikacji przechowywanych danych lub zakłócić je poprzez spam. Powyższy problem został zaadresowany poprzez rozwiązanie Masked Authenticated Messaging (MAM) [56]. MAM to protokół komunikacyjny, który z wykorzystaniem technik kryptograficznych dokonuje uwierzytelnienia przepływu danych. Stanowi integralną część protokołu IOTA odpowiedzialną za proces przesyłania i odczytywania wiadomości z wykorzystaniem splotu. MAM zapewnia integralność przesyłanych wiadomości oraz możliwość weryfikacji ich nadawcy. Każda przesyłana na nowy adres wiadomość wykorzystuje formę transakcji splotu. Każda wiadomość jest powiązana z kolejnymi wysyłanymi wiadomościami [56]. Każda transakcja *n* ma wskaźnik do transakcji *n+1*, jednakże nie posiadają one wiedzy o lokalizacji transakcji *n-1*.

MAM wykorzystuje schemat podpisu Merkle (Merkle Singature Scheme, MSS) [58]. Schemat ten jest oparty o drzewo Merkle (Merkle Hash Tree, MHT) oraz jednorazowe podpisy (One-Time Signatures, OTS) [59]. Schemat MSS jest odporny na atak z wykorzystaniem komputera kwantowego. Dodatkowo cechuje się krótkim czasem generacji i walidacji podpisu

oraz wysokim poziomem siły kryptograficznej. MSS pozwala na wykorzystanie jednego publicznego klucza do weryfikacji wielu OTS. Każda wiadomość drzewa skrótów (rysunek 10) jest podpisywana z wykorzystaniem schematu OTS i reprezentuje liście L1-L4. Wartość korzenia jest osiągana poprzez zastosowania funkcji skrótu do adresów [60]. Każde drzewo może wytworzyć liczbę wiadomości równą liczbie liści MHT. Każdy liść zawiera pełny adres następnego drzewa, który jednocześnie stanowi wskaźnik na następną wiadomość.

MAM posiada trzy tryby pracy:

- publiczny,
- prywatny,
- ograniczony.

W trybie publicznym korzeń MSS stanowi adres transakcji w splocie. W trybie prywatnym korzeń posiada skrót adresu określony jako *H(root)*, gdzie *H(.)* oznacza jednokierunkową funkcję skrótu. W trybie ograniczonym, wsad każdej z wiadomości jest dodatkowo zaszyfrowany i może być odszyfrowany jedynie przez posiadacza prawidłowego klucza [61]. Atakujący próbując odszyfrować wsad transakcji utworzonej w trybie prywatnym będzie miał dostęp do identyfikatora kanału, który jest skrótem kryptograficznym klucza kanału. Posiadanie jedynie wartości skrótu klucza nie daje odpowiedniej ilości informacji niezbędnych do odtworzenia wykorzystanego klucza kryptograficznego, co jednocześnie oznacza brak możliwości odszyfrowania wsadu tej transakcji. W trybie ograniczonym odbiorca wykorzystuje wartość korzenia drzewa do obliczenia adresu transakcji, co pozwala na wyszukania zamaskowanych wiadomości. Odszyfrowanie wiadomości wymaga wykorzystania wartości korzenia oraz klucza *sideKey* [59].

Wiadomości wysyłane z wykorzystaniem MAM to transakcje, w których nie zachodzi transfer środków. Transakcje te stanowią taką samą część splotu, jak pozostałe transakcje. Wiadomości te stanowią część rozproszonego rejestru i zwiększają bezpieczeństwo sieci poprzez zwiększenie całkowitej wagi sieci. Integralność zawartych w nich danych jest gwarantowana poprzez własności sieci. Wiadomości te są zazwyczaj potwierdzane w sposób pośredni.

MAM pozwala na wytworzenie prywatnego kanału komunikacji w publicznej sieci, jaką jest IOTA. Jedynie strony posiadające wiedzę na temat poprawnego adresu kanału posiadają dostęp do tych danych [62]. Ze względu na prywatny charakter rejestru sieci IOTA, ochrona wsadu tych wiadomości oraz wskaźników na następną transakcję pozwoliła na wytworzenie

takiego chronionego kanału komunikacyjnego. Dla przypadków użycia wymagających ograniczenia liczby uprawnionych do dostępu stron, a także do zarządzania prawami dostępu w przyszłości, w szczególności do odwoływania dostępu do danych, został przewidziany tryb ograniczony. Dodatkowe szyfrowanie wsadu pozwala na dostęp do danych jedynie stronom, które posiadają dodatkowy klucz kryptograficzny niezbędny do ich odszyfrowania.

Definicja 4.2.

Trajt to złożoną z trzech trytów jednostka danych, gdzie każdy tryt może przyjąć jedną z trzech wartości: 0, 1 lub 2, co czyni trajt podstawową jednostką alokacji w określonych systemach kodowania lub przetwarzania informacji.

Użytkownicy sieci IOTA mogą uzyskać dostęp do swojego konta poprzez hasło zwane ziarnem. Hasło to stanowi losowo wygenerowane 81 trajtów danych. Oznacza to, iż istnieje 27^{81} możliwych do wygenerowania ziaren. Dodatkowo w sieci IOTA stosowana jest idea kluczy prywatnych i publicznych. Klucz prywatny jest obliczany poprzez obliczanie skrótu ziarna z dodanym indeksem adresu. Indeks adresu to dowolna dodatnia liczba całkowita. Klucze publiczne to adresy, które zostały wyznaczone na podstawie wartości skrótu powiązanego klucza prywatnego. Adres posiada tokeny, które mogą być wykorzystane jako wejście transakcji lub jej wyjście. Oznacza to wydanie lub otrzymanie tokenu. Suma środków ze wszystkich adresów (kluczy publicznych) użytkownika jest obliczana, stanowiąc saldo jego konta [63].

Protokół IOTA wykorzystuje schemat jednorazowych popisów Winternitza [64] (Winternitz One-Time Signature, WOTS), który wykorzystuje podpisy oparte o skróty [63], przy czym jest odporny na ataki z wykorzystaniem komputerów kwantowych [65] i jest szybszy niż kryptografia oparta o krzywe eliptyczne[21] [66]. Wykorzystanie WOTS w IOTA niesie ze sobą także zagrożenia. Wraz ze wzrostem liczby transakcji transferu środków wykonywanych z danego adresu, maleje jego bezpieczeństwo. Jeżeli użytkownik często wykorzystuje ten sam klucz publiczny, dostarcza danych pozwalających na analizę generowanych przez niego

[21] Zgodnie z pracą [192] krzywa $E(F_{p^n})$, gdzie $p > 3$ jest liczbą pierwszą a n jest liczbą naturalną jest silna kryptograficznie, gdy spełnione są warunki:

1. $n = 1$ lub n jest liczbą pierwszą
2. $\#E(F_{p^n}) = kr$, gdzie $r > 2^{150}$ jest liczbą pierwszą oraz $(r, p) = 1$, zaś k jest małą liczbą naturalną $(1 \leq k \leq 2^8)$
3. $(p^n)^l \not\equiv 1 \bmod r$ dla $1 \leq l \leq 10$
4. Jeśli $n > 1$, to współczynniki a, b w równaniu krzywej nie należą do F_p^{75}
5. $d_{def} > 40$

przepływów. Analiza taka może pozwolić na odtworzenie historii oraz salda użytkownika, podobnie jak ma to miejsce w sieciach takich jak Bitcoin czy Ethereum [67]. Zaleca się wykorzystywanie nowych adresów do otrzymywania płatności. W [67] zostało także zaprezentowane rozwiązanie tego problemu z wykorzystaniem mikserów transakcji[22].

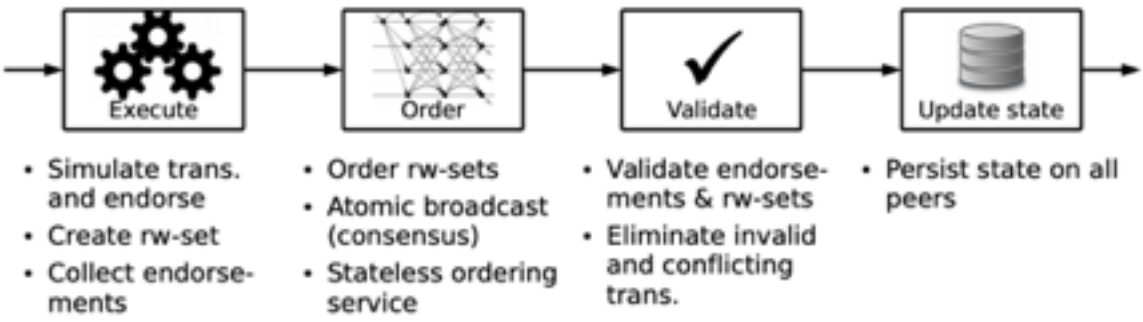
Powyższy problem istnieje, ponieważ za każdym razem, kiedy użytkownik przesyła tokeny ze swojego konta, zatwierdza daną transakcję ze swojego adresu. W tym celu wykorzystuje swój klucz prywatny. Za każdym razem, kiedy tworzony jest podpis, ujawniana jest połowa klucza prywatnego powiązanego z adresem wykorzystanym do wysłania tokenów [65]. Jeżeli użytkownik ponownie wykorzysta adres do wysłania kolejnych transakcji wychodzących, to znaczna część jego klucza prywatnego zostanie ujawniona, co pozwoli atakującemu na podszycie się pod jego tożsamość i kradzież środków. W aktualnej wersji protokołu IOTA ta słabość jest eliminowana poprzez przeszukiwanie splotu w poszukiwaniu transakcji wykonywanych z takich zużytych adresów. Jeżeli adres zostanie uznany za zużyty, nie będzie możliwe otrzymywanie na niego nowych środków. Należy jednak mieć na uwadze, iż po wykonaniu migawki rejestru, historia transakcji zostanie usunięta. Oznacza to, że nie będzie możliwe zweryfikowanie historii transakcji dla tego adresu i w konsekwencji może on zostać ponownie wykorzystany do otrzymywania środków.

[22] Mikser transakcji to narzędzie kryptograficzne używane w sieciach blockchain do zwiększenia prywatności poprzez ukrywanie źródła i przeznaczenia środków, polegające na agregacji wielu transakcji od różnych użytkowników w jedną, większą transakcję, co utrudnia śledzenie indywidualnych przepływów finansowych.

4.3. Blockchain prywatny – Hyperledger Fabric

W podrozdziale przedstawiono specyfikację rozwiązania Hyperledger Fabric. Opisano jego zdolność do wykonywania rozproszonych aplikacji, ze szczególnym naciskiem na określenie szczegółów architektury systemu opartego na modelu wykonuj-zamawiaj-waliduj.

Hyperledger Fabric to rozproszony system operacyjny dla blockchain z uprawnieniami, który wykonuje rozproszone aplikacje napisane w ogólnych językach programowania, takich jak Go, Java czy Node.js. W bezpieczny sposób śledzi historię wykonywania w strukturze danych o charakterze replikowalnego rejestru pozwalającego jedynie na dołączanie nowych wpisów i nie posiada powiązanej kryptowaluty.



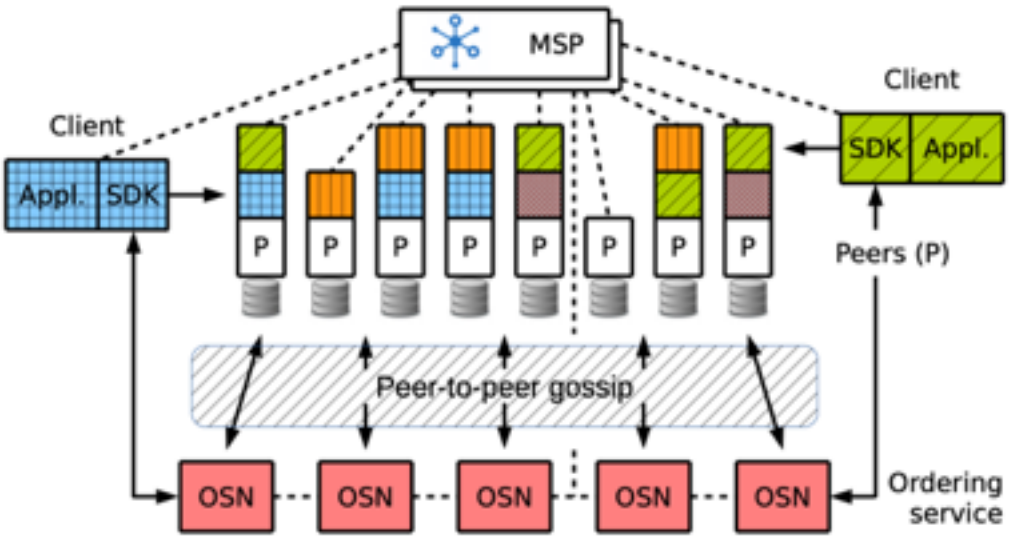
Rysunek 11 Architektura wykonuj-zamawiaj-waliduj dla Fabric [68]

Fabric wykorzystuje architekturę wykonuj-zamawiaj-waliduj (Rys. 11), co oznacza iż nie wykorzystuje standardowego dla rozwiązań blockchain schematu zamawiaj-wykonuj. Typowa rozproszona aplikacja w Fabric składa się z dwóch części:

- Smart kontraktu (*chaincode*), który stanowi kod programowy implementujący logikę aplikacyjną i jest realizowany w trakcie fazy wykonania. Chaincode stanowi centralną część rozproszonej aplikacji w Fabric i może zostać wytworzony przez niezaufanych deweloperów. Istnieje dedykowany kontrakt do zarządzania systemem blockchain i utrzymywania jego parametrów nazywany *system chaincode*.
- Polityki rekomendacji, która jest oceniana w fazie walidacji. Polityki rekomendacji nie mogą być wybierane ani modyfikowane przez niezaufanych deweloperów aplikacji. Polityki te stanowią statyczne biblioteki do walidacji transakcji w Fabric, mogąc być parametryzowane przez chaincode. Tylko wyznaczeni administratorzy mają uprawnienia do modyfikacji polityk rekomendacji poprzez funkcje zarządzania systemem. Typowa polityka rekomendacji pozwala chaincode określić

rekomendatorów dla transakcji w postaci zbioru węzłów koniecznych do akceptacji. Wykorzystuje monotoniczne funkcje logiczne na zbiorach, takie jak np. trzy z pięciu lub $(A \wedge B) \vee C$. Niestandardowe polityki rekomendacji mogą implementować dowolną logikę.

Klient wysyła transakcje do węzłów określonych przez politykę rekomendacji. Każda transakcja jest następnie wykonywana prze określone węzły, a jej wyjście jest zapisywane. Krok ten nosi także nazwę rekomendowania. Po wykonaniu, transakcja przechodzi do fazy kolejkowania, która wykorzystuje dołączany mechanizm konsensusu pozwalający na wytworzenie ostatecznej sekwencji rekomendowanych transakcji, pogrupowanych w bloki. Są one następnie rozsyłane do wszystkich węzłów, przy opcjonalnej pomocy szeptaczy. Fabric kolejkuje wyjścia transakcji połączone z zależnościami stanu wyznaczonymi podczas fazy wykonania. Jest to podejście odmienne od standardowej aktywnej replikacji [69], która kolejkuje wszystkie wejścia transakcji. Każdy z węzłów waliduje zmiany stany wynikające z rekomendowanych transakcji, biorąc pod uwagę ustaloną politykę rekomendacji oraz zwięzłość wykonania w fazie walidacji. Wszystkie węzły walidują transakcję w tej samej kolejności i weryfikują jej deterministyczność. Fabric wprowadza tu nowy paradygmat hybrydowej replikacji wykorzystujący model Bizantyjski, który stanowi połączenie replikacji pasywnej i aktywnej.



Rysunek 12 Sieć Fabric wykonująca kilka chaincode, zainstalowana na wybranych węzłach zgodnie z przyjętą polityką [68]

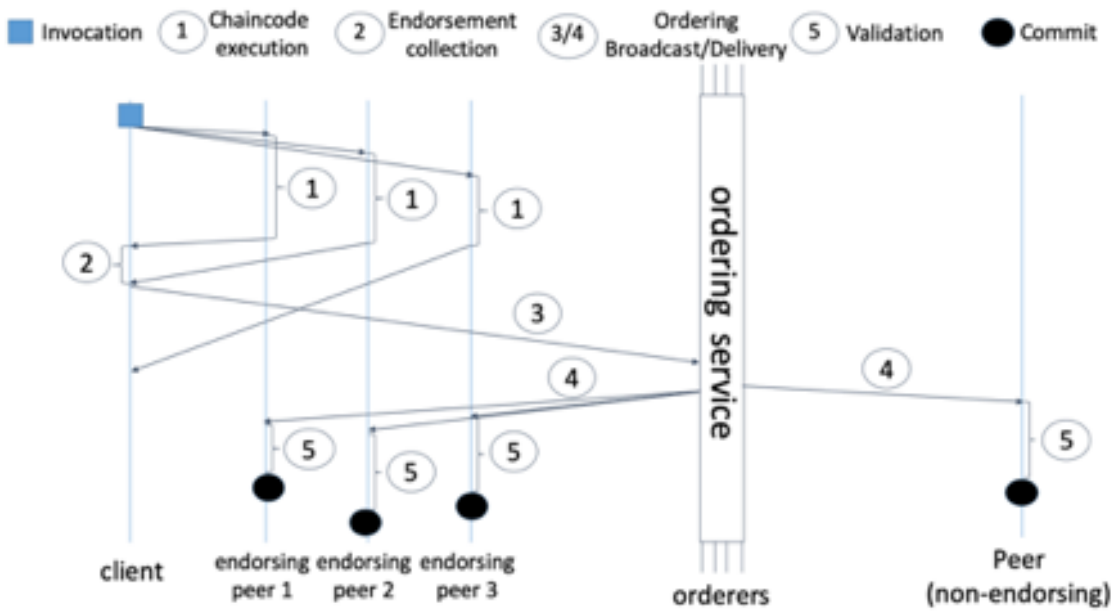
Blockchain Fabric składa się ze zbioru węzłów, które formują sieć (Rys. 12). Ponieważ Fabric jest blockchainem z uprawnieniami, wszystkie węzły które tworzą sieć mają swoją tożsamość dostarczoną przez modularny serwis dostarczający informacje o członkostwie (MSP – membership service provider). Węzły w sieci Fabric mogą przyjąć jedną z poniższych trzech ról:

- Klienci (clients) przesyłają propozycje transakcji do wykonania, pomagają przeprowadzić fazę wykonania i ostatecznie rozsyłają transakcje do kolejkowania.
- Węzły (peers) wykonują propozycje transakcji i walidują transakcje. Wszystkie węzły przechowują rejestr blockchain – strukturę danych pozwalającą jedynie na dodawanie nowych wpisów, przechowującą wszystkie transakcje w postaci łańcucha skrótów, wraz z ich stanem (state) – zwięzłą reprezentacją ostatniego stanu rejestru. Nie wszystkie węzły wykonują wszystkie propozycje transakcji. Ich podzbiór, nazywany węzłami rekomendującymi, realizuje wykonanie zgodnie z polityką chaincode, do którego należy transakcja.
- Węzły usługi zamawiającej (OSN – Ordering Service Nodes) to węzły które wspólnie tworzą usługą zamawiającą. Usługa ta odpowiada za ustalenie ostatecznej kolejności wszystkich transakcji w blockchain Fabric. Każda transakcja zawiera aktualizacje stanu i zależności obliczone podczas fazy wykonania wraz z podpisami kryptograficznymi rekomendujących węzłów. Zamawiający nie znają aktualnego stanu aplikacji i nie uczestniczą ani w wykonaniu, ani w walidacji transakcji. Taki wybór projektowy pozwala na zastosowania modułowego konsensusu i łatwą jego wymianę.

Sieć Fabric umożliwia wsparcie wielu sieci blockchain przyłączonych do tej samej usługi zamawiającej. Każdy z takich blockchainów jest nazywany kanałem i może posiadać różne węzły pełniące role członków. Kanały mogą być wykorzystywane do dzielenia stanu sieci blockchain, ale konsensus pomiędzy kanałami nie jest koordynowany i całkowita liczba transakcji w każdym z kanałów jest odmienna od pozostałych. Niektóre wdrożenia uznają, że wszyscy zamawiający są zaufani, co pozwala na implementację polityk dostępu do kanałów dla węzłów. W kolejnych punktach dokumentu kanały zostaną jedynie zasygnalizowane, a opis będzie się koncentrował na sieci z jednym kanałem.

W fazie wykonania klient podpisuje i przesyła propozycję transakcji do jednego lub wielu rekomendujących w celu jej wykonania. Każdy chaincode niejawnie określa zbiór rekomendujących poprzez ustanowioną politykę rekomendacji. Propozycja zawiera tożsamość

zgłaszającego klienta, wsad transakcji w formie operacji do jej wykonania, parametrów i identyfikatora chaincode, wartości nonce do jednokrotnego wykorzystania przez każdego klienta (np. licznik albo wartość losowa) oraz identyfikatora transakcji obliczonego na podstawie identyfikatora klienta i wartości nonce. Rekomendujący wykonują symulację propozycji poprzez wykonanie operacji na określonym chaincode, który został zainstalowany w blockchain. Chaincode jest uruchamiany w kontenerze Docker izolowanym od głównego procesu rekomendującego.



Rysunek 13 Wysokopoziomowy przepływ transakcji w Fabric [68]

Propozycja jest symulowana w oparciu o lokalny stan blockchain rekomendującego, bez dokonywania synchronizacji z innymi węzłami. Rekomendujący nie umieszczają wyników symulacji w stanie rejestru. Stan blockchain jest utrzymywany poprzez PTM (peer transaction manager) w formie wersjonowanego rejestru klucz-wartość, w którym kolejne aktualizacje wartości kluczy mają rosnące numery wersji. Stan wytworzony przez dany chaincode jest ograniczony wyłącznie do tego chaincode i nie może być osiągany ani bezpośrednio ani przez inny chaincode. Należy zaznaczyć, iż chaincode nie jest przewidziany do przechowywania lokalnego stanu w kodzie programu, a jedynie do przechowywania stanu blockchain, do którego dostęp jest możliwy poprzez operacje GetState, PutState oraz DelState. Posiadając odpowiednie uprawnienia, chaincode może pozwolić na dostęp innego chaincode do jego stanu, ale tylko w ramach tego samego kanału.

W wyniku przeprowadzonej symulacji, każdy z rekomendujących wytwarza wartość writeset, składającą się z aktualizacji stanu wytworzonego podczas symulacji (np. zmodyfikowane klucze wraz z ich nowymi wartościami) oraz readset reprezentującym wersję wszystkich zależności symulacji propozycji (np. wszystkie klucze odczytywane podczas symulacji wraz z numerami ich wersji). Po wywołaniu symulacji, rekomendujący podpisuje kryptograficznie wiadomość nazywaną rekomendacją, która zawiera readset oraz writeset (wraz z metadanymi takimi jak ID transakcji, ID rekomendującego oraz podpis rekomendującego) i odsyła je do klienta w postaci odpowiedzi na propozycję. Klient zbiera rekomendacje, dopóki nie zostaną spełnione założenia polityki rekomendacji dla tego chaincode. W szczególności wymaga to wykonania przez wszystkich rekomendujących, wskazanych przez politykę, takich samych wyników egzekucji (np. identycznych zbiorów writeset i readset). Następnie klient przystępuje do utworzenia transakcji i przekazuje ją do usługi zamawiającej.

Rekomendujący wykonują symulację propozycji bez synchronizacji z innymi rekomendującymi, zatem może się zdarzyć tak, że dwaj rekomendujący dokonają wykonania propozycji na różnym stanie rejestru i wytworzą różne wartości wyjściowe. Dla standardowych polityk rekomendacji wymaga się, aby wielu rekomendujących uzyskało takie same wyniki. W przypadku występowania wysokiej ilości operacji wykonujących dostęp do takich samych kluczy, powoduje to brak możliwości spełnienia wymagań polityki rekomendacji przez niektórych klientów. Jest to problem nowy w stosunku do problemu replikacji baz danych z wykorzystaniem oprogramowania pośredniczącego [70]. W konsekwencji prowadzi to do założenia, że żaden z węzłów nie jest zaufany do prawidłowego wykonania w blockchain.

Wprowadzone w Fabric rozwiązania pozwalają na uproszczenie architektury i są adekwatne dla typowych zastosowań blockchain. Jak pokazał przykład Bitcoin, rozproszone aplikacje mogą być sformułowane w taki sposób, że operacje wymagające dostępu do tego samego stanu są redukowane lub eliminowane kompletnie w przypadku normalnego wykonania. W przypadku Bitcoin dwie operacje, które modyfikują ten sam obiekt nie są dozwolone, ponieważ jest to atak typu double-spending [3]. Wykonanie transakcji przed fazą kolejkowania jest krytyczne dla niedeterministycznych chaincode. Chaincode w Fabric posiadający niedeterministyczne operacje stanowi zagrożenie dla żywotności jedynie własnych operacji, ponieważ klient mógł na przykład nie zebrać wystarczającej liczby rekomendujących. Jest to podstawowa przewaga nad architekturą typu zamawiaj-wykonuj, gdzie niedeterministyczne operacje powodują wystąpienie niespójności w stanie węzłów.

Tolerowanie niedeterministycznych wykonań pozwala także na wyeliminowanie ataków typu DoS realizowanych z niezaufanego chaincode, ponieważ rekomendujący może po prostu anulować wykonanie zgodnie z lokalną polityką, jeżeli tylko pojawi się podejrzenie ataku DoS. Nie spowoduje to zagrożenia dla spójności systemu. Tego rodzaju jednostronne zignorowanie wykonania nie jest możliwe w architekturze zamawiaj-wykonuj.

Po zebraniu przez klienta odpowiedniej liczby rekomendacji dla danej propozycji transakcji, klient kompiluje transakcję, która zawiera wsad (payload) składający się z operacji na chaincode wraz z parametrami, metadane, oraz zestaw rekomendacji. Następnie wysyła ją do serwisu zamawiającego. Faza kolejkowania zapewnia ustalenie ostatecznej kolejności transakcji w ramach danego kanału, wykorzystując do tego celu atomowe procesy, które zapewniają równoczesne rozsyłanie rekomendacji [71] i ustanawianie konsensusu, nawet w przypadku błędów w kolejności. Serwis zamawiający grupuje transakcje w bloki, tworząc połączoną hashami sekwencję bloków zawierających transakcje. Taki sposób grupowania zwiększa wydajność protokołu rozsyłania, korzystając z dobrze znanej metody tolerowania błędów w rozproszonych systemach komunikacyjnych.

Interfejs usługi zamawiającej obsługuje jedynie dwa następujące rodzaje operacji wywoływanych przez węzły i niejawnie parametryzowane przez identyfikator kanału:

- *broadcast(tx)*: Klient wywołuje tą operację do rozesłania arbitralnej transakcji *tx*, która zazwyczaj zawiera wsad transakcji oraz podpis klienta w celu dalszego rozpowszechnienia.
- $B \leftarrow deliver(s)$: Klient wywołuje tą operację w celu pozyskania bloku B o nieujemnym numerze sekwencyjnym *s*. Blok zawiera listę transakcji $[tx_1, \dots, tx_k]$ oraz połączoną hashami wartość h reprezentującą blok o numerze sekwencyjnym $s - 1$, np. $B = ([tx_1, \dots, tx_k], h)$. Klient może wywołać tą operację wielokrotnie i za każdym razem zwracany jest ten sam blok, o ile jest dostępny. Węzeł dostarcza blok B o numerze sekwencyjnym *s*, jeżeli otrzyma *B* po raz pierwszy po wywołaniu *deliver(s).*

Serwis zamawiający zapewnia całkowite uporządkowanie dostarczanych bloków w ramach jednego kanału. Proces kolejkowania pozwala na zapewnienie poniższych cech bezpieczeństwa dla każdego z kanałów:

- *Zgodność*. W przypadku, gdy dwa bloki, B i B', zostaną dostarczone do poprawnych węzłów sieci z tym samym numerem sekwencyjnym s, wówczas oba bloki muszą być identyczne, czyli $B = B'$.

- *Integralność łańcucha skrótów*. Jeżeli poprawny węzeł dostarczy blok B z numerem sekwencyjnym s i inny poprawny węzeł dostarczy blok $B' = ([tx_1, \dots, tx_k], h')$ o numerze $s + 1$, to $h' = H(B)$ gdzie $H(x)$ oznacza kryptograficzną funkcję skrótu.
- *Brak przeskakiwania*. Jeśli poprawny węzeł dostarczy blok B o numerze sekwencyjnym s większym od 0, wtedy każdy węzeł p musi dostarczyć wszystkie poprzednie bloki o numerach od 0 do $s - 1$. Zapewnia to, że węzły posiadają pełną i ciągłą historię łańcucha bloków bez brakujących segmentów.
- *Bez kreacji*. Gdy poprawny węzeł dostarcza blok B z numerem sekwencyjnym s, każda transakcja tx zawarta w B musi już być wcześniej rozesłana przez jakiegoś klienta. Oznacza to, że wszystkie transakcje w bloku są wynikiem zatwierdzonych operacji, a nie są tworzone arbitralnie przez węzły.

W celu zapewnienia żywotności, usługa zamawiająca może wspierać co najmniej poniższą właściwość:

- *Poprawność*. Jeżeli klient wywołał $broadcast(tx)$, to każdy poprawny węzeł dostarcza blok B, który zawiera tx oraz pewien numer sekwencyjny.

Hyperledger Fabric zezwala, aby każda indywidualna implementacja kolejkowania realizowała żywotność i bezstronność zgodnie z wymaganiami klienta.

Sieć blockchain może zawierać dużą liczbę węzłów, lecz tylko niewielka ich część jest odpowiedzialna za implementację usługi zamawiającej. Hyperledger Fabric umożliwia konfigurację wbudowanej usługi szeptającej (ang. gossip), która efektywnie rozpowszechnia bloki dostarczane przez usługę zamawiającą do wszystkich węzłów. Implementacja tej usługi jest skalowalna i niezależna od konkretnej implementacji usługi zamawiającej, co umożliwia współpracę z różnymi mechanizmami konsensusu, takimi jak BFT czy CFT, zapewniając modułowość i elastyczność architektury sieci blockchain.

Usługa zamawiająca może także przeprowadzać kontrolę dostępu w celu potwierdzenia, że klient jest uprawniony do rozsyłania wiadomości oraz otrzymywania bloków w ramach danego kanału.

Brak przechowywania stanu blockchain oraz walidacji wykonania transakcji w usłudze zamawiającej, to podstawowe cechy architektury Hyperledger Fabric. Pozwalają one na zbudowanie systemu blokchain kompletnie niezależnego od algorytmu konsensusu oraz wykonania i walidacji. Pozwala to na zachowanie modularności konsensusu oraz o ile to

możliwe, na utworzenie ekosystemu protokołów konsensusu implementujących mechanizm zamawiający. Integralność łańcucha skrótów oraz wiązanie bloków są wykorzystywane jedynie w celu weryfikacji integralności sekwencji bloków przez węzły w sposób jak najbardziej wydajny. Rozwiązanie to wyklucza także konieczność ochrony przed duplikacją transakcji. Upraszcza to implementację usługi zamawiającej a zduplikowane transakcje są odfiltrowywane przez węzły podczas sprawdzenia zapisu-odczytu walidacji transakcji.

W fazie walidacji bloki są dostarczane do węzłów bezpośrednio przez usługę zamawiającą lub z wykorzystaniem szeptaczy. Nowy blok stanowi wejście fazy walidacji, która składa się z trzech następujących po sobie kroków:

- Ewaluacja polityki rekomendacji jest przeprowadzana równolegle dla wszystkich transakcji w ramach bloku. Ewaluacja to zadanie VSCC (validation system chaincode) – statycznej biblioteki, która stanowi część konfiguracji blockchain i jest odpowiedzialna za walidację rekomendacji w stosunku do zastosowanej w chaincode polityki rekomendacji. Jeżeli rekomendacja nie jest spełniona, transakcja zostaje oznaczona jako nieprawidłowa, a jej efekty są odrzucane.
- Weryfikacja konfliktów odczytu-zapisu jest wykonywana sekwencyjnie dla wszystkich transakcji w bloku. Dla każdej transakcji wykonywane jest porównanie wersji kluczy w polu readset z tymi w aktualnym stanie rejestru – przechowywanymi lokalnie przez węzeł. Weryfikacja ma na celu potwierdzenie, że wartości te są takie same. Jeżeli wersje są różne, transakcja jest oznaczana jako nieprawidłowa, a jej efekty są odrzucane.
- Faza aktualizacji rejestru jest uruchamiana jako ostatnia. Podczas jej trwania blok jest dodawany do lokalnie przechowywanego rejestru, a stan blokchain jest aktualizowany. W szczególności dodawanie bloku do rejestru wymaga spełnienia poprzednich dwóch kroków a efekt ich wykonania jest przechowywany w postaci maski bitowej określającej, czy transakcje są prawidłowe w ramach danego bloku. Takie podejście wymusza wykonanie rekonstrukcji stanu w późniejszym czasie. Dodatkowo, wszystkie aktualizacje stanu są wykonywane poprzez zapisanie wszystkich zestawów klucz-wartość w writeset lokalnego stanu.

Domyślne VSCC w Fabric pozwala na realizację monotonicznych wyrażeń logicznych na zbiorze rekomendujących, konfigurowanych w kodzie kontraktu. Ewaluacja VSCC weryfikuje, czy zbiór węzłów wyrażony poprzez poprawne podpisy na rekomendujących transakcję spełnia założone wyrażenie. Różne polityki VSCC mogą być konfigurowane statycznie.

Rejestr Hyperledger Fabric zawiera wszystkie transakcje, także te które zostały uznane za nieważne. Powyższe wynika z przyjętej architektury rozwiązania. Usługa zamawiająca jest agnostyczna w stosunku do stanu chaincode i odpowiada za wytworzenie łańcucha bloków. Proces walidacji jest wykonywany przez węzły w konsensusie dokonywanym po tej fazie. Cecha ta jest konieczna w przypadku niektórych przypadków użycia wymagających śledzenia nieprawidłowych transakcji dla dokonywanych audytów. Blockchainy takie jak Bitcoin i Etherum nie oferują takiej funkcjonalności – przechowują one jedynie informacje o transakcjach prawidłowych. W wyniku wykorzystania systemu przywilejów w Fabric, istnieje możliwość wykrycia klientów próbujących wykonać atak DoS poprzez zalanie sieci nieprawidłowymi transakcjami. Jednym z możliwych do zastosowania podejść jest wykorzystanie czarnej listy takich klientów, zgodnie z przyjętą polityką bezpieczeństwa. Poszczególne wdrożenia mogą także implementować opłaty transakcyjne w celu obciążania klientów za każdą wywołaną transakcję, co spowoduje wzrost kosztów wykonania ataku DoS.

Fabric pozwala na zastosowanie elastycznych założeń na zaufanie i winę. W ogólnym przypadku, każdy klient jest uznawany za złośliwego lub bizantyjskiego. Węzły są grupowane w organizacje, a każda organizacja formuje jedną domenę zaufania taką, że każdy węzeł w tej domenie ufa wszystkim węzłom w tej organizacji, ale nie ufa żadnemu węzłowi spoza niej. Usługa zamawiająca uznaje wszystkie węzły i klientów za potencjalnie bizantyjskich.

Integralność sieci Fabric jest oparta na spójności usługi zamawiającej. Model zaufania usługi zamawiającej zależy bezpośrednio od jej implementacji. Fabric od wersji 1.06 wspiera scentralizowaną, jednowęzłową implementację wykorzystywaną zarówno w procesie rozwoju jak i testowania oraz usługę zamawiającą CFT wykonywaną na klastrze. Trzecia implementacja, to PoC BFT-SMaRt [72] pozwala na tolerancję do jednej trzeciej bizantyjskich OSN [73].

Należy zauważyć, że sieć Fabric oddziela model zaufania dla aplikacji od modelu zaufania konsensusu. Mianowicie, aplikacja rozproszona może definiować własne założenia zaufania, które są przekazywane za pośrednictwem polityki rekomendacji i są niezależne od założeń konsensusu realizowanych przez usługę zamawiającą.

4.4. Technologie zdecentralizowanego przechowywania danych

W niniejszym podrozdziale omówiono wyzwania technologii blockchain, koncentrując się na rosnących wymaganiach dotyczących przechowywania danych. Opisano, jak konwencjonalne sieci blockchain, zaprojektowane głównie do transferu wartości, nie realizują potrzeb nowoczesnych aplikacji generujących duże ilości danych. W odpowiedzi na te wyzwania, przedstawiono rozwiązania takie jak IPFS i Arweave, które umożliwiają zdecentralizowane przechowywanie danych przy zachowaniu ich dostępności i integralności. Przeanalizowano mechanizmy tych systemów, w tym sposób adresowania danych za pomocą identyfikatorów CID, struktury oraz protokoły komunikacyjne umożliwiające efektywne wyszukiwanie i dystrybucję danych.

Wraz ze wzrastającym wykorzystaniem technologii blockchain w tworzeniu nowych aplikacji i usług zaobserwowano nowe przypadki użycia, które nie mogły być obsłużone z wykorzystaniem istniejących rozwiązań blockchain. Największą słabością blockchain jest wysoki koszt przechowywania danych. Protokoły te zostały zaprojektowane w celu transferu środków i nie są przygotowane do składowanie ilości danych, jakie są generowane przez nowoczesne aplikacje, takie jak streaming wideo czy sieci społecznościowe. Potrzeba wytworzenia protokołu pozwalającego na zdecentralizowane przechowywanie danych, gwarantującego ich dostępność i integralność, co zaowocowało powstaniem rozwiązań takich jak IPFS [74] oraz Arweave [75].

IPFS – InterPlanetary File System stanowi protokół tworzący metodę przechowywania i udostępniania plików w oparciu o adresowanie zawartością. Protokół wykorzystuje szereg rozwiązań pozwalający na adresowanie, wyszukiwanie i udostępnianie informacji. Do najważniejszych komponentów należy zaliczyć [74]:

- Mechanizm adresowania treści z wykorzystaniem CID (Content Identifiers).
- Mechanizm linkowania treści z wykorzystaniem skierowanych grafów acyklicznych Merkle (Merkle Directed Acyclic Graph, DAG).
- Mechanizm wyszukiwania treści z wykorzystaniem rozproszonych tablic skrótów (Distributed Hash Tables, DHT).

Adresowanie zasobów w sieci IPFS odbywa się z wykorzystaniem identyfikatorów CID. Są one generowane z wykorzystaniem kryptograficznych funkcji skrótu i mogą być interpretowane jako etykiety danych pozwalające na ich zaadresowanie w sieci IPFS. CID nie zawiera informacji o położeniu treści w sieci, jednakże może być rozważany jako adres zasobu

[76]. CID może zostać wytworzone z wykorzystaniem wielu funkcji skrótu. Najczęściej wykorzystywana jest SHA2-256 [77], ale może to być także Blake2 [78] a także MD5 [79]. Zastosowanie funkcji skrótu do generowania CID pozwala na uzyskanie unikalnych identyfikatorów dla danych, a także pozwala na ich wzajemne łączenie. Wyróżnia się dwie wersje CID:

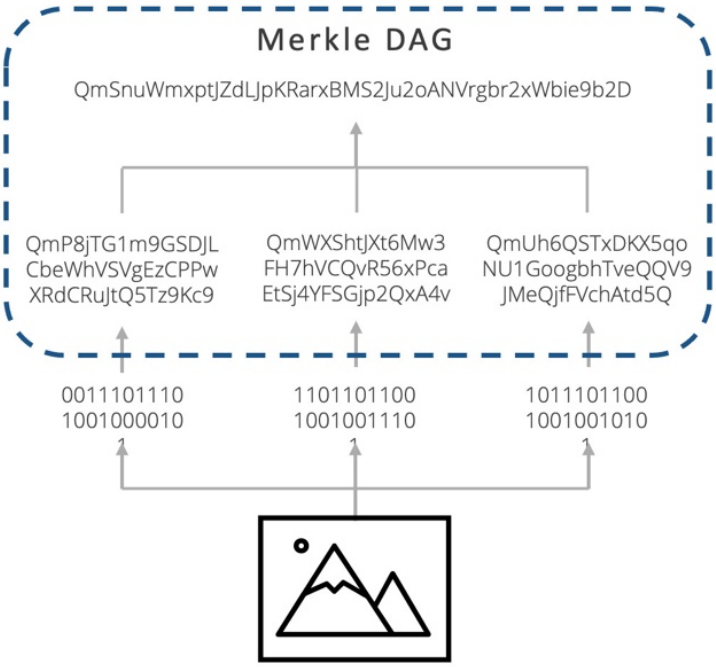
- Wersja 0 wykorzystująca kodowanie base58, np.: *QmbWqxBEKC3P8tqsKc98xmWNzrzDtRLMiMPL8wBuTGsMnR*
- Wersja 1 w której dodano prefiks oznaczający wybrany typ kodowania, wersję CID oraz format adresowanej zawartości, np.: *bafybeigdyrzt5sfp7udm7hu76uh7y26nf3efuylqabf3oclgtqy55fbzdi*

Wersja 1 pozwala na dodawanie nowych informacji do CID, co umożliwia dostosowywanie identyfikatorów do zmieniających się wymagań i do nowych przypadków użycia [76].

Zaadresowanie treści z wykorzystaniem CID nie jest wystarczające, aby mogły one być dostępne w sieci. Do tego celu niezbędne jest ich linkowanie, które ze względu na rozproszony charakter IPFS jest dokonywane z wykorzystaniem specyficznej struktury – skierowanych grafów acyklicznych Merkle [80]. Struktura ta przypomina drzewo Merkle [58], jednakże nie ma narzuconego wymogu zbalansowania gałęzi. IPFS wykorzystuje DAG do reprezentowania struktur folderów i plików. Każdy węzeł DAG posiada unikalny identyfikator, który jest skrótem kryptograficznym zawartości tego węzła. Powyższe pozwala na łatwą reprezentację zawartości i jej identyfikację jedynie za pomocą wartości CID.

Wykorzystywany w IPFS skierowany graf acykliczny posiada następujące właściwości:

- Dysponuje unikalnym identyfikatorem.
- Jest ustrukturyzowany w taki sposób, że każdy liść grafu jest zawarty w nadrzędnym DAG, każdy węzeł stanowi korzeń DAG i jest zawarty w nadrzędnym DAG [81].
- Jest niezmienny. Wprowadzenie zmiany w zawartych w nim treściach powoduje wygenerowanie nowego identyfikatora, wpływającego na wartości, efektem czego jest zmiana identyfikatorów wszystkich nadrzędnych DAG, co skutkuje utworzeniem zupełnie nowej struktury.
- Węzły posiadające taki sam identyfikator reprezentują ten sam DAG.
- Integralność całej zawartości sieci może zostać zweryfikowana z wykorzystaniem CID.



Rysunek 14 Struktura Merkle DAG [76]

Proces tworzenia nowego DAG jest realizowany poprzez podzielenie zawartości na bloki po 256 kB [82]. Na podstawie bloków generowane są liście grafu, zawierające CID treści. W kolejnych krokach powstają kolejne poziomy DAG aż do jego korzenia (Rysunek 14). Wprowadzony podział na bloki pozwala na ich udostępnienia z różnych lokalizacji, zwiększając decentralizację. Ze względu na wykorzystanie CID, weryfikacja integralności odbywa się w sposób zautomatyzowany. Jeżeli IPFS byłby wykorzystywany do przechowywania zbioru plików, to w przypadku aktualizacji jednego z nich jedynie CID zmienionego pliku zostanie zaktualizowany. Pozwala to także na redukcję ilości danych przechowywanych w sieci oraz zmniejsza wymagania narzucane na transfer.

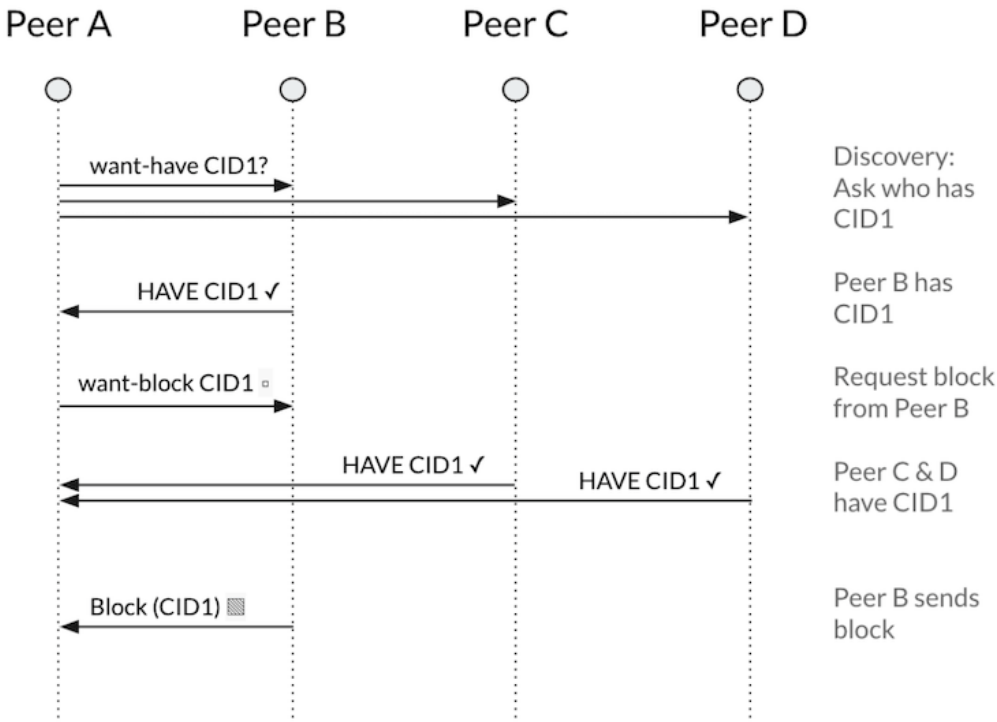
Wyszukiwanie zawartości w sieci IPFS jest oparte o rozproszone tablice skrótów DHT. Dodatkowo, w procesie wyszukiwania biorą też udział technologie Bitswap [83] oraz libp2p [84]. Rozproszona tablica skrótów (DHT) to baza danych w postaci klucz-wartość, wykorzystywana do wyszukiwania treści w sieci IPFS. Proces ten jest stosowany do określenia węzłów sieci udostępniających daną treść. Tablica skrótów jest przechowywana na wszystkich węzłach sieci IPFS [85]. DHT są popularnym rozwiązaniem w systemach wymiany danych peer-to-peer, a jednym z najbardziej rozpowszechnionych rozwiązań jest Kademlia DHT [86]. Innymi szeroko wykorzystywanymi rozwiązaniami są Coral DSHT [87] oraz S/Kademlia DHT [88]. Coral DSHT stanowi rozszerzenie Kademlia DHT zawierające liczne usprawnienia w

zakresie wydajności i przechowywania danych. S/Kademlia DHT jest rozszerzeniem skupionym na bezpieczeństwie danych.

Sieć IPFS wykorzystuje rozproszone tablice skrótów do odnajdowania węzłów sieci przechowujących bloki poszukiwanej zawartości oraz do określenia ich fizycznej lokalizacji w sieci. Wykorzystany algorytm routingu oparty o identyfikatory węzłów został opisany w [89] i nie będzie przedmiotem dalszych rozważań tej pracy.

Pozyskanie zawartości na podstawie ustalonej lokalizacji jest możliwe z wykorzystaniem protokołu Bitswap [83]. Protokół ten jest odpowiedzialny za transfer wiadomości określających listę żądanych bloków danych oraz ich transferu pomiędzy węzłami sieci. Sieć IPFS wysyła żądanie bloków do Bitswap, a ten pozyskuje je z fizycznej sieci. Proces pozyskiwania danych przebiega zgodnie z następującymi krokami:

1. Główny CID jest dodawany do listy żądań, stanowiącej listę CID wszystkich bloków, które dany węzeł chce pozyskać.
2. Żądany blok jest pozyskiwany z połączonych węzłów sieci z wykorzystaniem rozgłoszenia żądania. Jeżeli żaden z połączonych węzłów nie posiada wskazanego bloku, przejdź do kroku 3, w przeciwnym przypadku przejdź do kroku 4.
3. Przeszukaj DHT w poszukiwaniu identyfikatorów węzłów posiadających żądany blok.
4. Jeżeli żądany blok jest dostępny, węzeł przesyła dane. W przeciwnym przypadku węzeł otrzymuje dane pozyskane z DHT dotyczące identyfikatorów węzłów przechowujących daną zawartość. Węzły takie zostają dodane do aktualnej sesji połączenia.
5. Pozyskaj dane zawarte w głównym bloku. Analizując jego zawartość pozyskaj CID wszystkich bloków składających się na zawartość.
6. Pozyskane CID są dodawane do listy żądań, a algorytm wraca do punktu 1 i jest wykonywany aż do pobrania wszystkich bloków danej zawartości.



Rysunek 15 Pozyskiwanie zawartości z wykorzystaniem protokołu Bitswap [90]

Protokół Bitswap dokonuje wyboru preferowanych węzłów sieci, z których zostaną pozyskane dane. Wybór ten jest określany na podstawie opóźnienia w komunikacji pomiędzy węzłami [83]. Im niższa wartość opóźnienia transmisji tym wyższa szansa na wybór danego węzła. Protokół ma możliwość priorytetyzacji jedynie na podstawie opóźnienia transmisji, ponieważ ze względu na zastosowanie adresacji z wykorzystaniem CID i idącej za tym ochronie integralności danych nie istnieje potrzeba zaufania do węzłów sieci. Bezpiecznym jest założenie, iż każdy węzeł udostępniający materiał o danym CID udostępnia jego właściwą formę.

Protokół Bitswap działa efektywnie, jeżeli węzły sieci biorą udział w udostępnianiu danych. W aktualnej implementacji IPFS węzły nie otrzymują nagrody za pozostawanie w gotowości do udostępnienia danych i sieć działa w oparciu o własną potrzebę węzłów do udostępniania przechowywanych na nich danych. Jako odpowiedź na ten problem, twórcy IPFS przedstawili własną kryptowalutę Filecoin [91], której celem jest wynagradzanie węzłów sieci za przechowywanie i udostępnianie danych.

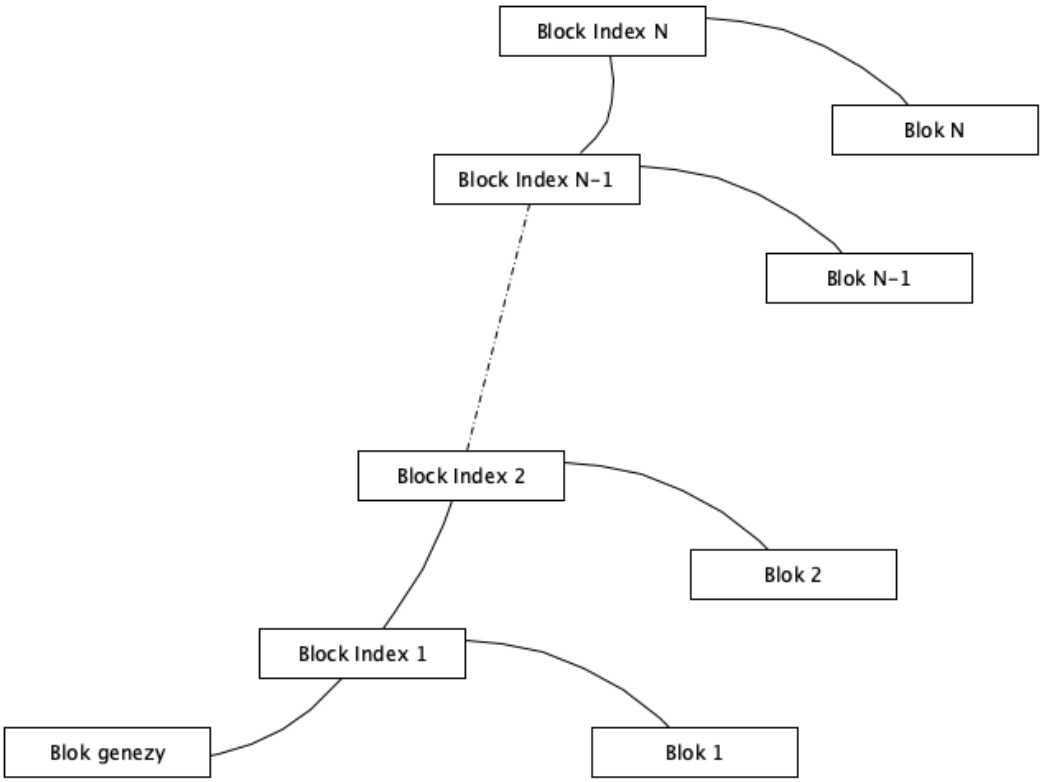
Implementacja programowa protokołów wykorzystywanych w sieci IPFS została wykonana w postaci zestawu bibliotek dystrybuowanych pod nazwą libp2p [92]. Do podstawowych zadań realizowanych przez libp2p należą:

- Zapewnienie uwierzytelnienia węzłów. Każdy węzeł wykorzystuje parę kluczy kryptograficznych wykorzystywanych do zabezpieczania komunikacji i uwierzytelniania węzłów sieci. Identyfikator węzła stanowi skrót kryptograficzny jego klucza publicznego zakodowany w taki sam sposób jak wersja 0 CID.
- Dostarczenie mechanizmów adresowania. Biblioteka pozwala na komunikację w różnych typach sieci (np. TCP/IP lub P2P). Powyższe jest osiągane poprzez wykorzystanie tzw. multiadresów, których struktura jest zależna od wykorzystywanego protokołu komunikacyjnego. Przykładowo może to być */ip4/1.2.3.4/udp/5678* lub */p2p/QmX...Ac1*.
- Transfer danych pomiędzy węzłami. Libp2p dostarcza szeregu protokołów, które są wykorzystywane do implementacji trybu nasłuchu, kiedy to połączenia od innych węzłów są akceptowane oraz trybu wybierania, kiedy to węzeł otwiera nowe połączenie do nasłuchujących węzłów.
- Wyszukiwanie zawartości poprzez implementację Kademlia DHT.
- Obsługa routingu dla węzłów. Libp2p implementuje routing wiadomości pomiędzy węzłami sieci z wykorzystaniem algorytmu Kademlia DHT.
- Obsługa protokołu PubSub do tworzenia zaawansowanych aplikacji aktualizujących stan w czasie rzeczywistym [93].
- Konfiguracja NAT (Network Address Translation) do obsługi połączeń przychodzących [94].

Alternatywnym wobec IPFS rozwiązaniem zdecentralizowanego przechowywania danych jest Arweave [75]. W odróżnieniu od IPFS, które stanowi jedynie usługę zdecentralizowanego przechowywania i udostępniania danych w oparciu o sieć peer-to-peer, Arweave jest rozwiązaniem korzystającym z technologii blockchain. Sieć pozwala użytkownikom na trwałe przechowywanie danych (do 200 lat) po wniesieniu jednorazowej opłaty transakcyjnej zależnej od rozmiaru przechowywanych danych i aktualnej pojemności sieci. Wykorzystana do przechowywania danych struktura drzewa Merkle [58] gwarantuje brak możliwości usunięcia danych składowanych w sieci, bez naruszenia jej integralności. W kolejnych akapitach tego rozdziału opisane zostały szczegóły techniczne tego rozwiązania.

Celem, jaki był postawiony przed Arweave było zapewnienie trwałego i skalowalnego przechowywania danych. Powyższe oznacza, iż Arweave nie definiuje bezpośrednio warstwy aplikacji, dostarczając jedynie narzędzi do budowy rozwiązań takich jak rozproszone bazy danych, czy też platformy obsługi smart kontraktów [95, 96]. Protokół wykorzystuje szereg

algorytmów i struktur danych, które łącznie stanowią konsensus proof of storage dla danych zgromadzonych w sieci [97].

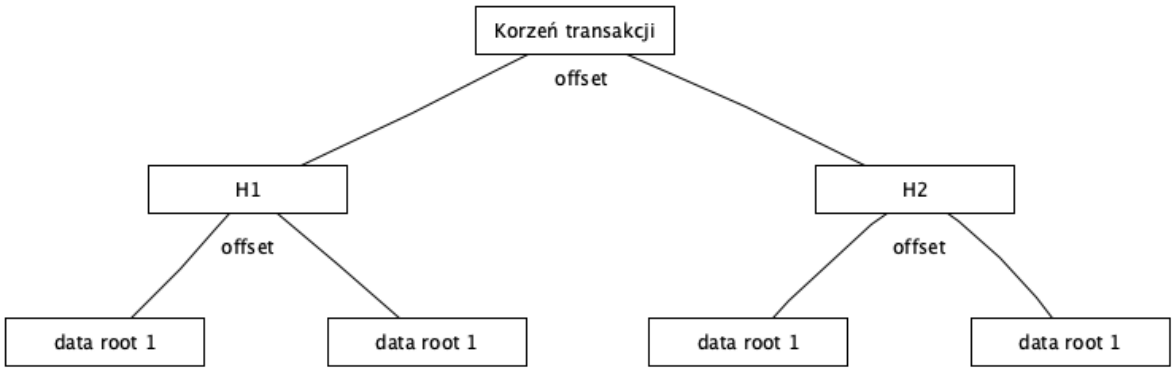


Rysunek 16 Indeksy bloków w postaci drzewa skrótów bloków sieci Arweave [97]

Podstawową strukturę danych Arweave tworzy Block Index (BI). Strukturę tą tworzy lista krotek, z których każda zawiera skrót kryptograficzny danego bloku, rozmiar splotu oraz identyfikator korzenia transakcji. Powyższa lista jest reprezentowana w postaci drzewa Merkle [98], którego korzeń określa najbardziej aktualny stan sieci. Składa się on z dwóch elementów – skrótu krotki określającej ostatni blok oraz poprzedniego indeksu bloku (BI). W procesie tworzenia nowych bloków następuje ich wiązanie z poprzednim blokiem. Dokonywane jest to poprzez umieszczenie wartości skrótu korzenia drzewa Merkle dla poprzedniego bloku w bloku aktualnie tworzonym. Zastosowanie powyższej konstrukcji pozwala węzłom w sieci Arweave na pełną walidację dowolnego bloku poprzedzającego aktualny blok, dla którego zweryfikowane zostały warunki konsensusu. Znacząco ogranicza to to nakład pracy niezbędny do walidacji poprawności poprzednich bloków, w szczególności wyklucza potrzebę pobierania danych i analizy ich nagłówków.

Podobnie jak ma to miejsce w przypadku IPFS, dane które są dodawane do sieci Arweave muszą zostać podzielone na części o rozmiarze 256 kB. Na podstawie wygenerowanych części

tworzone jest drzewo Merkle (data root), a jego korzeń jest dodawany do transakcji. Transakcja jest następnie podpisywana i wysyłana do sieci przez użytkownika dodającego dany plik. Każda transakcja jest reprezentowana przez wartość korzenia transakcji zapisaną w bloku, w którym została zawarta. Każdy blok zawiera szereg transakcji, dla których górnicy zatwierdzający blok obliczają wartość korzenia transakcji uwzględniając dane zawarte w drzewie Merkle danego pliku i umieszczają ją w nagłówku nowo tworzonego bloku.



Rysunek 17 Korzeń transakcji składający się z data root zawartych w transakcji [97]

Każdy z węzłów drzewa Merkle jest oznaczany z wykorzystaniem offsetu danych, które występują w jego sąsiedztwie. Etykiety te są wykorzystywane do tworzenia dowodów pozwalających na weryfikację istnienia fragmentów pliku w danych lokalizacjach. Taki rodzaj drzewa Merkle jest nazywany niezbalansowanym, ponieważ liczba liści dla każdej z gałęzi nie musi być równa. Sieć Arweave implementuje proces określany jako Succint Proof of Access (SpoA). Jego zadaniem jest udowodnienie stronie *B*, iż strona *A* przechowuje wskazane dane w pewnej lokalizacji. Przeprowadzenie takiego dowodu wymaga posiadania przez strony tego samego korzenia drzewa Merkle dla danego zbioru danych oraz przeprowadzenie trójfazowego procesu zakładającego fazę wyzwania, konstrukcji dowodu oraz jego weryfikacji.

W fazie wyzwania, strona *B* wysyła do *A* offset wyzwania. Jest to indeks bajtu, do którego dostęp strona *A* ma udowodnić. W fazie konstrukcji dowodu, *A* dokonuje przeszukania drzewa Merkle w celu odnalezienia fragmentu danych odpowiadającego przesłanemu offsetowi. *A* pobiera całą zawartość tej części danych, oblicza jej skrót kryptograficznych i odszukuje rodzica tego węzła w drzewie Merkle. Rodzic posiad swój offset oraz skróty dla dwóch węzłów będących jego dziećmi. Wszystkie wykorzystywane skrótu mają 256 bitów. Wartość rodzica jest dodawana do konstruowanego dowodu. *A* przeszukuje drzewo w górę, dodając do dowodu wartości wszystkich węzłów aż do korzenia przeszukiwanego drzewa. Następnie *A* wysyła tą ścieżkę dostępu do danych stronie *B*, dołączając do dowodu zawartość żądanej części danych.

W fazie weryfikacji, *B* potwierdza otrzymany od *A* dowód oraz część danych. Weryfikuje węzły w otrzymanym dowodzie przechodząc przez ścieżkę i weryfikując skróty z przesłanego dowodu ze skrótami znajdującymi się w drzewie. Następnie oblicza skrót kryptograficzny dla otrzymanych danych i weryfikuje czy jest on zgodny ze skrótem zamieszczonym w liściu wskazywanym przez dowód. Jeżeli wszystkie skróty są zgodne, to dowód uznaje się za poprawny. Jeżeli weryfikacja jest błędna dla jakiegokolwiek ze skrótów uznaje się, że dowód był niepoprawny i otrzymane dane odrzuca się.

Sieć Arweave została zaprojektowana w taki sposób, aby każdy z jej uczestników posiadał dostęp do proporcjonalnej liczby replik splotu. Powyższe narzuca wymagania na zachowanie tych samych danych w każdej ich kopii przechowywanej w sieci. Rzeczone jest osiągane poprzez zastosowanie schematu pakowania. Pakowanie ma na celu zapewnić, iż górnicy sieci nie będą mieli możliwości sfałszowania dowodów SPoA potwierdzających przechowywanie wielu unikalnych replik tych samych danych. Sieć wykorzystuje w tym celu algorytm RandomX [99], który dokonuje konwersji pomiędzy regularnymi kawałkami danych a ich formą opakowaną. Powyższy proces jest realizowany przez górników sieci i zakłada wykonanie poniższych kroków:

- Obliczany jest skrót kryptograficzny offsetu danego kawałka, korzenia transakcji oraz adresu. Wynikiem jest klucz wykorzystywany do pakowania.
- Uzyskany klucz stanowi źródło entropii dla algorytmu generującego brudnopis[23] o rozmiarze 2MB.
- Pierwsze 256 kB uzyskanego brudnopisu jest wykorzystane do zaszyfrowania kawałka danych z wykorzystaniem sieci Feistela [100]. Szyfrogram jest umieszczany w nowym kawałku.

W przypadku gdy koszt pakowania kawałków danych jest większy niż koszt przechowania opakowanych kawałków w danym czasie pomiędzy kolejnymi odczytami, górnik może przechowywać kawałki danych zamiast dokonywać ich opakowywania na żądanie. Powyższe ma na celu zwiększeni wydajności pracy sieci. W celu podjęcia właściwej decyzji, górnik weryfikuje poniższą nierówność:

$$c_s(data) \cdot t < c_p(data)$$

[23] Brudnopis w sieci Arweave to obszar pamięci wykorzystywany do szyfrowania danych przed ich opakowaniem.

gdzie:

- c_s oznacza koszt przechowywania danych,
- c_p oznacza koszt opakowywania,
- t oznacza średni czas pomiędzy odczytem danych.

Na koszt opakowywania składa się koszt energii elektrycznej wykorzystanej w procesie obliczeń oraz amortyzacja wykorzystywanego w tym celu sprzętu. Koszt przechowywania stanowi koszt zakupu przestrzeni dyskowej, jej amortyzacji oraz koszty bieżącego utrzymania, np. koszt energii elektrycznej. Sieć Arweave ustaliła wartość stosunku kosztów opakowywania do kosztów przechowywania na poziomie 19, na podstawie analiz i testów przeprowadzonych z wykorzystaniem zalecanych konfiguracji sprzętowych i programowych.

Proces wyznaczania klucza pakującego wykorzystuje algorytm RandomX [99], który został skonstruowany w taki sposób, aby utrudnić opracowanie akceleratorów sprzętowych pozwalających na znaczące przyspieszenie obliczeń. Algorytm jest zoptymalizowany pod kątem wykonywania na procesorach ogólnego przeznaczenia. Wykorzystuje w tym celu ulosowione wykonanie kodu oraz kilka technik znacząco zwiększających zapotrzebowanie na pamięć. Powyższe wskazuje, że jedynym sposobem na zwiększenie wydajności RandomX jest stosowanie bardziej wydajnych procesorów ogólnego przeznaczenia, co w najbliższym czasie pozwala na zachowanie ograniczonego wzrostu podaży mocy obliczeniowej w sieci Arweave.

Arweave wykorzystuje weryfikowalną funkcję opóźnienia (ang. Verifiable Delay Function VDF [101]). Zadaniem tej funkcji jest obliczeniowe zweryfikowanie upływu czasu pomiędzy pewnymi zdarzeniami. Funkcja VDF wykonuje n kroków w celu wytworzenie wartości wyjściowej, która może być w sposób efektywny potwierdzona [102]. Sieć Arweave wykorzystuje następującą rekurencyjną definicję funkcji VDF:

$$\begin{cases} VDF(n, seed) = SHA2_{256}(seed), & dla\ \ n = 1 \\ VDF(n, seed) = SHA2_{256}(VDF(n-1, seed)), dla\ n > 1 \end{cases}$$

Przytoczona konstrukcja zapewnia, że kolejny skrót może zostać wygenerowany jedynie w przypadku, gdy poprzedni został już policzony. Powyższe uniemożliwia zrównoleglenie obliczeń i tym samym zapewnia odpowiednio długi czas wykonania. Poprawnie wygenerowana wartość *VDF(k,seed)* określana jest jako punkt kontrolny. Wartość ta potwierdza upływ co najmniej 1 sekundy od momentu otrzymania ziarna do wygenerowania punktu kontrolnego. Proces obliczania wartości VDF ma złożoność obliczeniową *O(n)*. Weryfikacja może zostać wykonana w czasie *O(n/p)*, pod warunkiem, że weryfikator będzie posiadał dostęp do punktów

kontrolnych powiązanych z danym VDF. Obliczenia mogą wtedy zostać wykonane z wykorzystaniem *p* równoległych wątków.

Sieć Arweave implementuje proces oparty na opisywanym we wcześniejszych akapitach SPoA, który jest wykorzystywany do potwierdzenia posiadania przez stronę udowadniającą pewnej liczby replik danych. Proces ten jest określany jako Succint Proofs of Replications (SPoRes) i umożliwia weryfikację posiadania replika danych przy ich minimalnym transferze i narzucie związanym z wymaganymi obliczeniami.

W procesie SPoRes strona *A* twierdzi, że przechowuje *n* kopii danego zbioru danych, podczas gdy strona *B* chce zweryfikować to twierdzenie. W celu rozpoczęcia procesu weryfikacji, *B* przekazuje do *A* wartość *d* będącą parametrem określającym trudność oraz pewne losowe ziarno *s*. A wykorzystuje otrzymane dane do wygenerowania łańcucha wartości VDF, co sekundę wysyłając wartości punktów kontrolnych wykorzystywanych do odblokowania *k* wyzwań SPoA. Każdorazowe pakowanie kawałków danych pozwala na rozwiązanie tych wyzwań, pozwalając na skonstruowanie przez *A* odpowiednich dowodów SPoA. Następnie dla każdego z tych dowodów obliczana jest wartość skrótu i porównywana z zadanym parametrem trudności *d,* gdzie $d \in (0, 2^{256})$. Jeżeli obliczony skrót dowodu jest większy od wartości *d*, wtedy jest on uznawany za rozwiązanie prawidłowe i przedstawiany *B*. *B* rejestruje czas, jaki zajęło dostarczenie prawidłowego rozwiązania przez *A*. Prawdopodobieństwo odnalezienia poprawnego rozwiązania w jednym przejściu jest równe:

$$p = \frac{2^{256} - d}{2^{256}}$$

Jeżeli *A* dysponowałoby *N* replikami pozwalającymi na wykonanie *k* przejść w czasie sekundy, to prawdopodobieństwo odnalezienia rozwiązania w tym czasie jest równe:

$$p_2 = 1 - (1 - p)^{kN}$$

Czas, który strona *A* poświęca na dostarczenie dowodu posiadania określonej liczby replik danych może być modelowany z wykorzystaniem zmiennej rozkładu geometrycznego, w którym prawdopodobieństwo powodzenia wynosi p_2. Zmienna ta jest zależna od wartości *d*, *k* oraz *N*. Przyjmując, że strona *B* oczekuje otrzymania dowodu posiadania wystarczającej liczby replik co 120 sekund, wyznacza wartość *d* w taki sposób, aby strona *A* była w stanie w tym czasie dostarczyć odpowiedni dowód. Oznacza to, iż wartość $p_2 = \frac{1}{120}$, co dla przypadku ogólnego daje:

$$p_2 = 1 - \left(\frac{d}{2^{256}}\right)^{kN}$$

Jeżeli strona *A* posiada wystarczającą liczbę replik danych, wtedy będzie dostarczała stronie *B* dowody w oczekiwanym czasie. Strona *B* weryfikuje kolejne przesyłane dowody oraz czas ich generowania, żeby potwierdzić generowanie przez *A* dowodów w granicach wyznaczonych 120 sekund przez odpowiednio długi czas. Jeżeli *A* będzie w stanie dostarczać dowody w wyznaczonym czasie przez okres obserwacji, to *B* może z dużym prawdopodobieństwem potwierdzić przechowywanie przez A właściwej liczby replik.

Niech strona *A* twierdzi, że przechowuje 20 replik danych, a *B* niech żąda przesyłania odpowiedniego dowodu co 120 sekund przez okres 2 tygodni. Łącznie *B* oczekuje otrzymania 10080 dowodów w badanym okresie. Prawdopodobieństwo tego, że *A* posiada jedynie 19 replik i jest w stanie dostarczyć odpowiednią liczbę dowodów jest równe:

$$p_2^* = 1 - \left(\frac{d}{2^{256}}\right)^{19k}$$

Wartość powyższego prawdopodobieństwa może zostać wyznaczona zgodnie z poniższym:

$$p_2 = 1 - \left(\frac{d}{2^{256}}\right)^{20k} = \frac{1}{120}$$

$$\left(\frac{d}{2^{256}}\right)^{20k} = \frac{119}{120}$$

$$\left(\frac{d}{2^{256}}\right)^{19k} = \left(\left(\frac{d}{2^{256}}\right)^{20k}\right)^{\frac{19}{20}} = \left(\frac{119}{120}\right)^{\frac{19}{20}} \approx 0{,}9921$$

$$p_2^* = 1 - \left(\frac{d}{2^{256}}\right)^{19k} \approx 0{,}0079$$

Na podstawie powyższego możliwe jest wyznaczenie oczekiwanego czasu generowania dowodu, w przypadku, gdy A posiada jedynie 19 replik danego zestawu danych. Niech X^* będzie zmienną losową o rozkładzie geometrycznym z parametrem p_2^*. Wartość ta może zostać wyznaczona zgodnie z poniższym:

$$E(X^*) = \frac{1}{p_2^*} \approx \frac{1}{0{,}0079} \approx 126{,}5823$$

Prawdopodobieństwo tego, że średni czas generowania dowodów będzie wynosił mniej niż 120 sekund może być oszacowane z wykorzystaniem centralnego twierdzenia granicznego [103]. Może zostać określone z wykorzystaniem następującego prawdopodobieństwa:

$$P\left(\frac{\sum X_i^*}{10080} \leq 120\right) = P\left(\frac{\sum X_i^*}{10080} \leq 126{,}5823 - 6{,}5823\right)$$

gdzie:

- $\sum X_i^*$to suma wyników z i prób, gdzie każda X_i^* jest liczbą prób potrzebną do osiągnięcia sukcesu w i-tej serii prób.
- $i \in \{1,2,\ldots,10080\}$.

Proces SPoRes znajduje swoje zastosowanie w operacji wydobywania bloków sieci Arweave. W procesie tym sieć odtwarza opisaną rolę strony B, podczas gdy górnicy wcielają się w rolę strony A. Ta część protokołu Arweave nie zostanie opisana w niniejszej rozprawie.

Wymienione w niniejszym rozdziale usługi zdecentralizowanego przechowywania danych – IPFS oraz Arweave należą do najbardziej popularnych w chwili powstania tej rozprawy. Innymi istotnymi z punktu widzenia udziału rynkowego usługami są:

- Filecoin
- Storj
- Sia

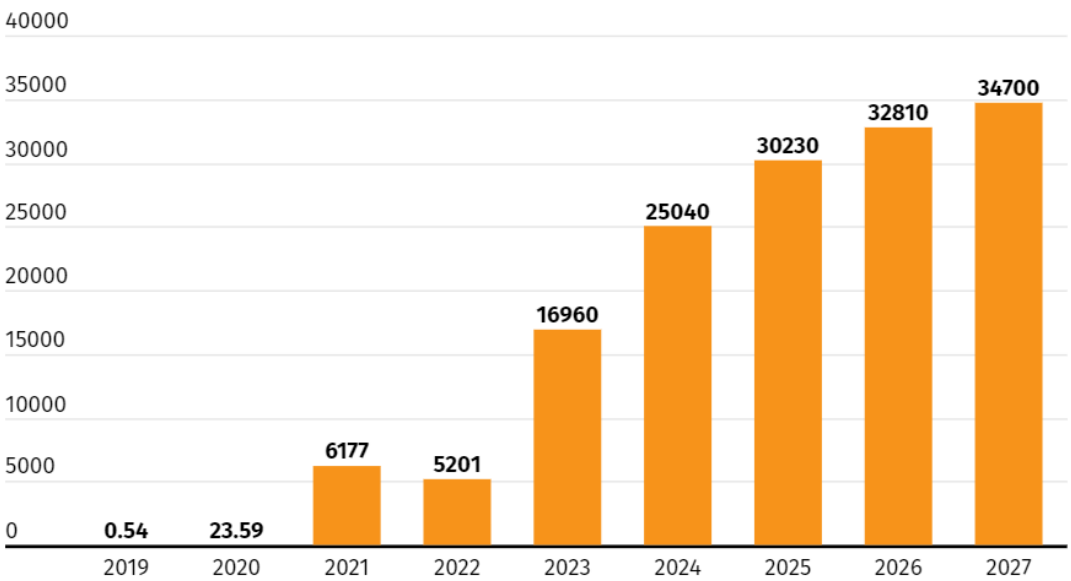
Filecoin [104] jest zdecentralizowanym systemem przechowywania danych opartym o sieć IPFS. Pozwala użytkownikom na otrzymywanie opłat za przechowywanie danych, determinując koszt w oparciu o podaż i popyt na rynku przechowywania danych. Storj [105] oferuje usługę przechowywania danych w chmurze stworzonej przez użytkowników tej zdecentralizowanej sieci. Dane przechowywane w Storj są szyfrowane na końcówkach sieci, implementując zasadę end-to-end-encryption. Ostatnia z usług - Sia [106] podobnie jak Storj przechowuje dane w sposób zdecentralizowany, na urządzeniach należących do użytkowników sieci. Dane są przechowywane w postaci zaszyfrowanej i rozproszone w sieci. Użytkownicy udostępniający przestrzeń dyskową otrzymują wynagrodzenie, podobnie jak ma to miejsce w przypadku Filecoin i Storj.

4.5. Zastosowania blockchain 2.0 i 3.0

W niniejszym podrozdziale omówiono rozwój i zastosowania technologii blockchain 2.0 i 3.0, ze szczególnym uwzględnieniem smart kontraktów i ich wpływu na nowoczesne usługi cyfrowe. Szczegółowo opisano funkcjonowanie zdecentralizowanych finansów (DeFi) i zdecentralizowanych autonomicznych organizacji (DAO), podkreślając ich znaczenie w automatyzacji i decentralizacji zarządzania oraz transakcji finansowych. Znaczącą część poświęcono także roli i specyfice tokenów niezamiennych (NFT), uwypuklając ich unikalność i zastosowanie w praktyce, w tym różnice pomiędzy standardami ERC-721 i ERC-1155, które wpływają na efektywność i bezpieczeństwo transakcji.

Projekty blockchain 2.0, na przykład Ethereum, mają na celu poszerzenie funkcjonalności obecnych systemów blockchain poprzez umożliwienie wykonywania kodu opracowanego przez użytkowników sieci. Pozwoliło to na znaczący rozwój możliwości wykorzystania sieci w stosunku do pierwszej generacji blockchain oferującej jedynie transfer wirtualnych środków. Swoboda definiowania smart kontraktów zaowocowała wytworzeniem szeregu nowych rozwiązań, wśród których można wyróżnić następujące najpopularniejsze przypadki użycia:

- Zdecentralizowane finanse (DeFi, Decentralized Finance) [107]
- Zdecentralizowane autonomiczne organizacje (DAO, Decentralized Autonomous Organization) [108]
- Kolekcjonerskie aktywa cyfrowe (NFT, Non Fungible Token) [109]



Rysunek 18 Rynek DeFi w mln USD [111]

Aplikacje zdecentralizowanych finansów (DeFi) to aplikacje o otwartym kodzie, wdrażane na publicznych sieciach blockchain. Aplikacje te w ostatnich latach przeżywają intensywny wzrost wartości przetwarzanych środków (Rysunek 18). Sieć blockchain jest wykorzystywana jako infrastruktura do przechowywania historii transakcji w sposób zapewniający ich integralność i audytowalność. W sieci tej implementowane są standaryzowane smart kontrakty, takie jak np. ERC-20 [110], które mają za zadanie dostarczyć aktywów bazowych dla warstwy aktywów danej sieci. Aplikacja DeFi jest wdrażana również w postaci smart kontraktu, który ma za zadanie dostarczenie logiki biznesowej i jej deterministyczne wykonanie. Do najpopularniejszych zastosowań DeFi należy zaliczyć:

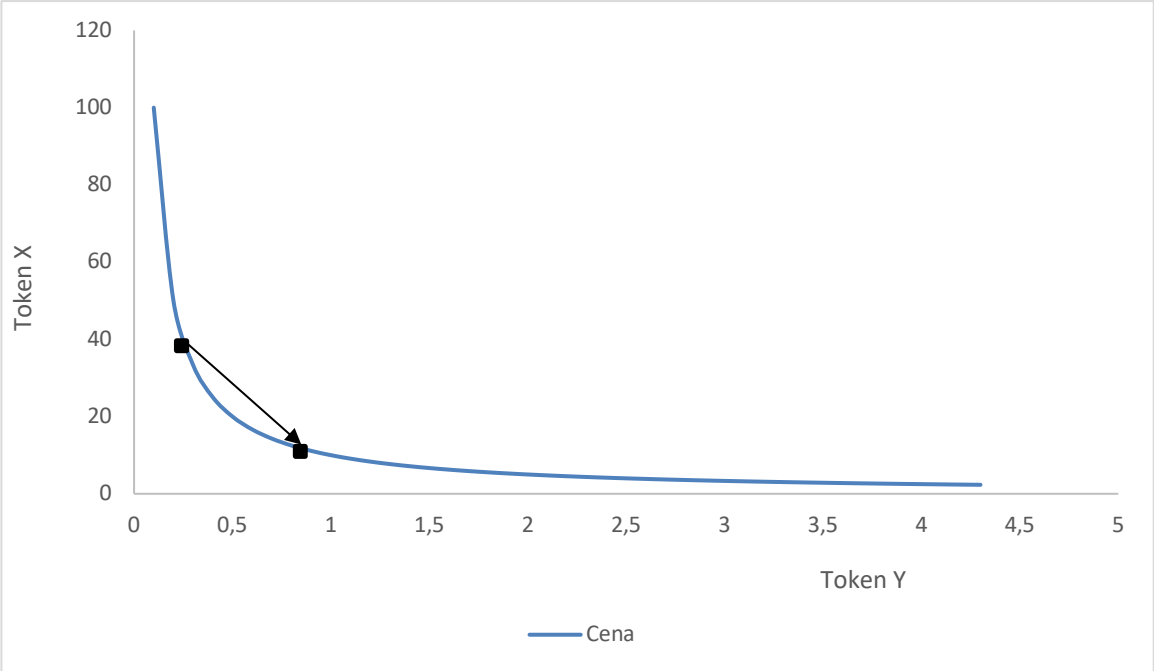
- Zdecentralizowane giełdy (DEX), np. Uniswap [112]
- Platformy pożyczkowe, np. Aave [113]
- Derywatywy na rynku kryptowalut, np. BitMEX [114]
- Zdecentralizowane ubezpieczenia, np. Nexus Mutual [115]

Niezależnie od wybranego przypadku użycia, platformy te posiadają podobnych aktorów realizujących działania w danym protokole. Należą do nich użytkownicy końcowi, wykorzystujący funkcjonalności platformy, dostawcy płynności gwarantujący odpowiedni kapitał do funkcjonowania platformy, arbitrzy posiadający możliwość przywrócenia funkcjonowania aplikacji wynikającego z zaistniałych w niej zdarzeń oraz twórcy platformy odpowiedzialni za zaprojektowanie, zaimplementowanie oraz bieżące utrzymanie danej aplikacji.

Przez wzgląd na ograniczenia wprowadzane przez sieci blockchain, a w szczególności wysoki koszt przechowywania danych, nie było możliwe efektywne zaimplementowanie giełdy opartej o centralny rejestr zleceń kupna i sprzedaży. Powyższe poskutkowało stworzeniem konstrukcji opartej o tzw. zautomatyzowanego animatora rynku (AMM, Automated Market Maker) [116]. AMM wyznacza cenę danej pary aktywów w oparciu o dostępną płynność. Niech x oznacza ilość tokenów X, a y ilość tokenów Y będących w dyspozycji AMM. Cena wymiany jest określana z wykorzystaniem następującego równania (Rysunek 19):

$$x \cdot y = k$$

gdzie k to stała zdefiniowana w ramach danego protokołu.



Rysunek 19 Schemat wyznaczania ceny dla pary tokenów

Zdecentralizowane platformy pożyczkowe bazują na istnieniu dostawców płynności, którzy są gotowi zablokować posiadane przez siebie środki w zamian za odsetki wypłacane w trybie ciągłym. Charakter sieci blockchain wyklucza tradycyjną formę pożyczania aktywów, w której pożyczający nie musi posiadać środków na pokrycie spłaty pożyczki. W przypadku zdecentralizowanych platform pożyczkowych, pożyczkobiorca musi zagwarantować zabezpieczenie pożyczki, wnoszone w postaci aktywów innych niż pożyczane, których wartość w momencie zaciągania pożyczki jest wyższa niż wartość pożyczanych tokenów. Wartość jest porównywana na podstawie wartości tokenów wyrażonych w dolarze amerykańskim, a wysokość pożyczki nie przekracza 70-90% wniesionego zabezpieczenia.

Zdecentralizowana autonomiczna organizacja (DAO) to system zbudowany w oparciu o blockchain, który pozwala użytkownikom na koordynację i zarządzanie działaniami określonymi przez zbiór reguł zaimplementowanych w publicznej sieci blockchain [117]. DAO nie jest terminem definiującym konkretne oprogramowanie, lecz określa zbiór aplikacji, które posiadają następujące cechy:

- Są zbudowane w oparciu o smart kontrakty wdrożone na zdecentralizowanej infrastrukturze. Kontrakty te są zarządzane zgodnie z zaimplementowanymi zasadami w sposób w pełni automatyczny.

- Nie są zależne od żadnych sił zewnętrznych. Nie ma potrzeby fizycznej interakcji z implementacją, jest ona uruchamiana w sposób automatyczny na węzłach sieci, w której DAO jest wdrożona.
- Są w pełni autonomiczne. Brak jest scentralizowanej kontroli podejmowanych decyzji.
- Wszystkie operacje są zautomatyzowane i wykonywane zgodnie z wdrożonym kodem.
- Posiadają mechanizm, który pozwala na modyfikację reguł organizacji.
- Kod jest publicznie dostępny i audytowalny, gwarantując transparentność organizacji i podejmowanych przez nią decyzji.
- Nie posiada ograniczeń geolokalizacyjnych co do jej działania.
- Posiada formę wyłącznie cyfrową, gdzie wszystkie reguły są zapisane i wykonywane przez odpowiednie smart kontrakty wdrożone w sieci blockchain.
- Zaimplementowane reguły są odporne na bezpośrednie ataki i nie mogą być łatwo zmieniane. Proces aktualizacji reguł jest zaimplementowany w kodzie smart kontraktów i jest niezmienny.
- DAO zapewnia interoperacyjność poprzez rozpoznawanie zdarzeń w sieci oraz bezpośrednie reagowanie na interakcję z jej kontraktami.
- Uczestnicy DAO mają zapewnioną anonimowość podobną do tej oferowanej przez bazowy blockchain.

W chwili powstawania niniejszej rozprawy za najbardziej obiecujące DAO były uznawane projekty RobotEra [118], Calvaria [119], Battle Infinity [120], Uniswap [121] oraz Sushiswap [122].

Tokeny niezamienne (Non Fungible Tokens) zostały wprowadzone przez ERC-721 [123] w roku 2018. Ich podstawowym celem jest reprezentowanie aktywów cyfrowych, które w odróżnieniu od kryptowalut, czy tokenów standardu ERC-20 stanowią indywidualne utwory, dla których istotne jest odróżnienie poszczególnych instancji. Podstawowym celem było umożliwienie obrotu dziełami cyfrowymi – obrazami, zdjęciami, muzyką czy wideo, ale także reprezentowanie biletów wstępu, cyfrowych identyfikatorów itp. [124]. NFT mogą także zostać wykorzystane do wytworzenie systemów uwierzytelniania użytkowników [125]. W chwili pisania tej rozprawy kapitalizacja NFT przekroczyła 70 miliardów dolarów.

W standardzie ERC-721 przewidziano ograniczone możliwości przechowywania danych w sieci blockchain. Powyższe wynika z wysokich kosztów ponoszonych za składowanie danych w sieci, co w przypadku aktywów cyfrowych nierzadko powodowałoby przekroczenie wartości danego aktywu i skutecznie uniemożliwiało swobodny obrót. W standardzie ERC-721 określono następujące interfejsy dla metadanych:

```
interface ERC721Metadata{
    function name() external view returns (string _name);
    function symbol() external view returns (string _symbol);
    function tokenURI(uint256 _tokenId) external view returns
(string);
}
```

Powyższy kod określa następujące składowe metadanych:

- *name* – nazwa danego tokenu, typ string
- *symbol* – symbol danego tokenu, typ string
- *tokenURI* – ujednolicony identyfikator zasobów (Uniform Resource Identifier [126]) zawierający ścieżkę do właściwego pliku metadanych, typ string

Właściwe dane cyfrowe będące przedmiotem NFT są umieszczone pod adresem wskazywanym przez *tokenURI*. Wskazywany plik najczęściej ma postać struktury JSON definiującej metadane w sposób bardziej szczegółowy, wraz ze wskazaniem fizycznej lokalizacji danego aktywu [127]. Metadane powinny być przechowywane w sposób gwarantujący integralność i dostępność danych. W szczególności powinien zostać wykorzystany jeden z systemów zdecentralizowanego przechowywania danych opisany w poprzednim rozdziale – Arweave lub IPFS. Przez wzgląd na brak możliwości modyfikacji metadanych NFT po jego wybiciu, wybór właściwej usługi przechowywania danych jest kluczowy do utrzymania wartości tego aktywu.

Tabela 3 Porównanie standardów ERC-1155 oraz ERC-721 [131]

Cecha	ERC-1155	ERC-721
Transfer tokenów	Możliwość transferu partii tokenów należących do różnych standardów (ERC-20, ERC-721) w ramach jednej transakcji	Transfer jedynie jednego tokenu w transakcji

Czas i koszt	Transferowania tokenów w partiach pozwala na obniżenie kosztu transakcji, pozwala także przyspieszyć transfer tokenów.	Możliwość transferu jedynie jednego tokenu w transakcji zwiększa koszty i czas wymagany do transferu.
Bezpieczeństwo	ERC-1155 implementuje mechanizmy wycofywania transakcji, co pozwala na odzyskanie tokenów w sytuacji dokonania transferu na nieprawidłowy adres.	Nie ma możliwości wycofania transakcji w sytuacji wykonania błędnego przelewu.
Wsparcie dla częściowo zamiennych tokenów	Tworzenie i przesyłanie	Brak wsparcia

Wzrost kosztów obsługi transakcji w sieci Ethereum spowodował powstanie standardu ERC-1155 [128], który eliminował wybrane problemy standardu ERC-721, takie jak ograniczenia na liczbę transferowanych tokenów, czy konieczność wdrażania nowego smart kontraktu dla każdego NFT. Wyeliminowanie tych ograniczeń pozwoliło na propozycje nowych przypadków użycia, takich jak handel energią elektryczną [129], czy podział zysków z wynajmu współdzielonego pojazdu [130].

5. Decentralizacja infrastruktury z wykorzystaniem DLT

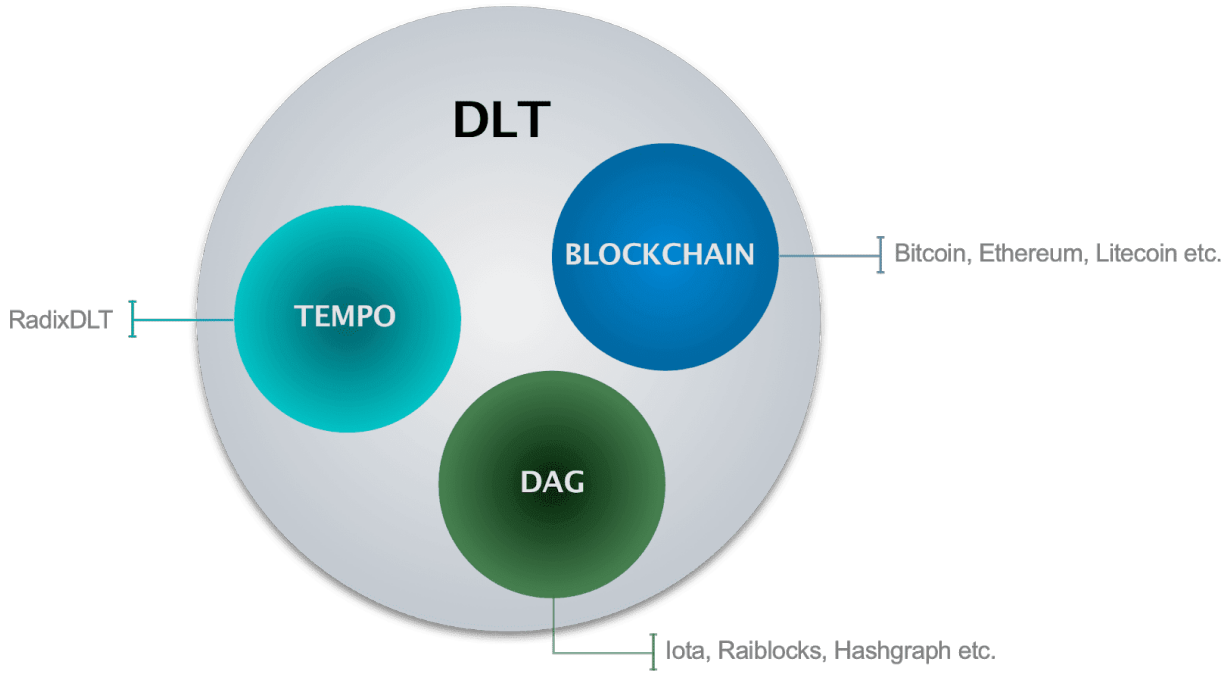
W rozdziale przedstawiono autorskie propozycje wykorzystania technologii Distributed Ledger Technology (DLT) w celu decentralizacji infrastruktury. Szczególną uwagę skupiono na trzech kluczowych aspektach: integralności, dostępności i obliczeniach rozproszonych. Analiza wskazuje, jak DLT może przyczynić się do zwiększenia bezpieczeństwa i efektywności systemów w każdym z tych obszarów. Zaprezentowane koncepcje stanowią solidne fundamenty dla przyszłych badań oraz praktycznych implementacji w różnych sektorach, podkreślając potencjalne korzyści płynące z decentralizacji procesów i danych.

5.1. Integralność

Podrozdział ten skupia się na analizie autorskich propozycji dotyczących integralności danych w kontekście zastosowania technologii rozproszonych rejestrów, takich jak blockchain i skierowane grafy acykliczne (DAG). W tej części opisano różne mechanizmy zabezpieczające, takie jak sumy kontrolne, kryptograficzne funkcje skrótu i kody MAC, które mają na celu ochronę danych przed nieautoryzowanymi zmianami. Podkreślono, że zastosowane w technologiach DLT metody są szczególnie odporne na ataki, co wynika z wymagań dotyczących konsensusu i intensywności obliczeń niezbędnych do modyfikacji danych. Ponadto, zaprezentowana została analiza autorskiego przypadku użycia systemu rozproszonego do monitorowania łańcucha dostaw produktów medycznych, ukazując przewagi takiego podejścia nad tradycyjnymi systemami scentralizowanymi.

Integralność danych to własność wykluczająca wprowadzenie do nich zmian w nieautoryzowany sposób [132]. Zapewniana jest poprzez zastosowanie mechanizmów wykrywania zmian, takich jak sumy kontrolne, kryptograficzne funkcje skrótu, czy też kody MAC [132]. W zależności od narzuconych wymagań, celem może być jedynie wykrycie błędu transmisji, który następnie może zostać skorygowany z wykorzystaniem kodu korekcji błędów lub retransmisja w celu wyeliminowania wcześniej powstałego błędu transmisji. Jeżeli wymaganiem jest ochrona integralności danych przed celową modyfikacją wprowadzoną przez aktorów wewnętrznych lub zewnętrznych, wtedy wykorzystuje się zaawansowane techniki kryptograficzne, takie jak jednokierunkowe funkcje skrótu, kody MAC (ang. Message Authentication Codes [133]), a także schematy podpisów cyfrowych. Powyższe techniki należą do dobrze zbadanych i szeroko wdrożonych w aktualnie stosowanych systemach informatycznych. Pomimo powyższego, sposób ich implementacji w systemie zazwyczaj pozwala na odnalezienie ścieżki ataku, która pozwala na naruszenie integralności danych [134].

Technologie rejestru rozproszonego, takie jak blockchain czy skierowane grafy acykliczne (DAG), są zaprojektowane w celu zachowania pełnej integralności danych. Sposób, w jaki dane są ze sobą powiązane ma na celu wykluczenie możliwości wprowadzenie modyfikacji zapisów wprowadzonych do sieci. Zmiana stanu sieci wymaga ogromnych zasobów obliczeniowych, a także osiągnięcia konsensusu z innymi uczestnikami. Sprawia to, że przeprowadzenie praktycznego ataku na rejestr rozproszony jest niemożliwe [136].

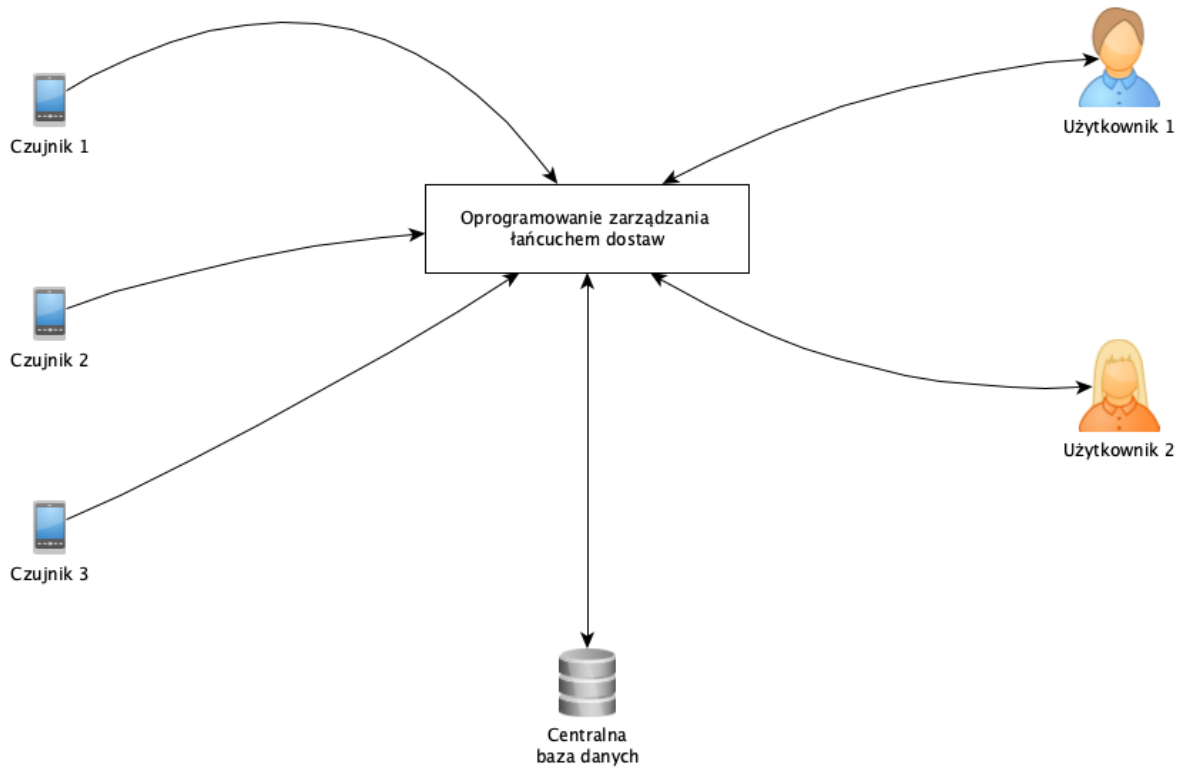


Rysunek 20 Technologie rejestru rozproszonego [135]

Takie cechy rejestry rozproszonego pozwalają na rozważenie jego zastosowania w obszarach życia gospodarczego i społecznego, które wymagają nadzwyczajnych rozwiązań do ochrony integralności danych. Przeanalizujmy szczególny przypadek użycia, polegający na monitorowaniu łańcucha dostaw dla produktów medycznych wymagających szczególnych warunków logistycznych, np. utrzymania kontrolowanej atmosfery. Analizowany scenariusz przewiduje, że firma *X* dokonuje transportu produktów medycznych z punktu *A* do punktu *B*. Ze względu na konieczność zapewnienia najwyższej jakości i bezpieczeństwa transportowanych produktów, niezbędne jest utrzymanie ich w temperaturze od 2 do 5 stopni Celsjusza przez cały czas transportu.

Załóżmy, że powyższy przypadek użycia zostanie obsłużony z wykorzystaniem systemu scentralizowanego. System taki może zostać wdrożony w jednej organizacji, która będzie posiadała pełną kontrolę nad wszystkimi jego elementami składowymi. W szczególności system ten będzie posiadał następujące elementy:

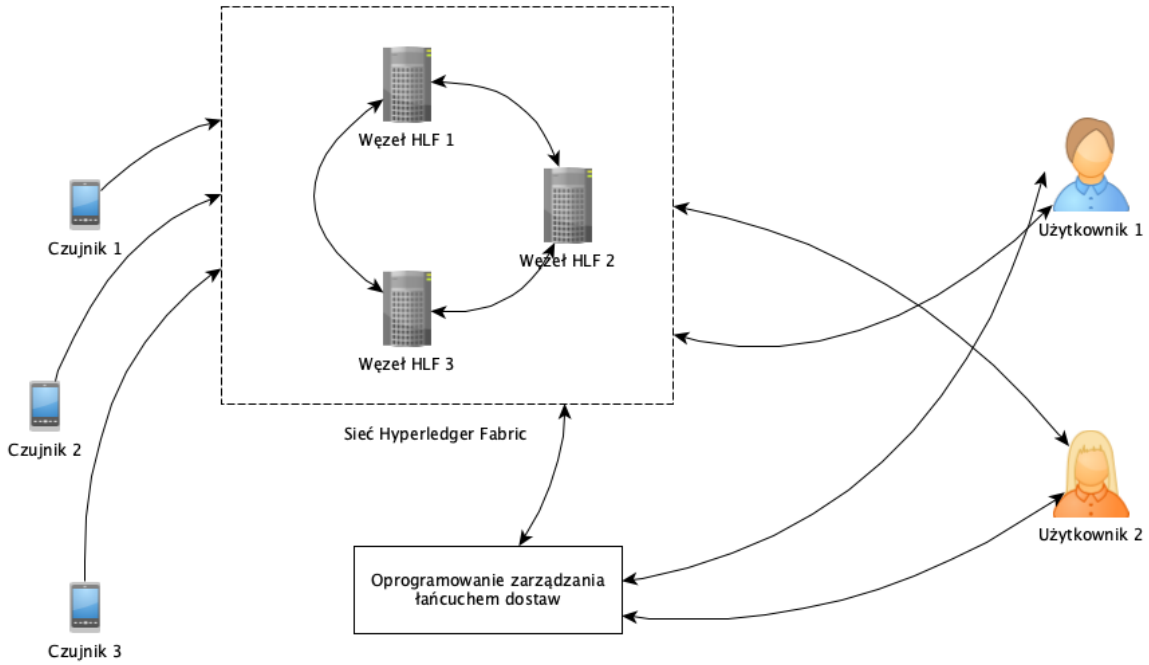
- Czujniki IoT umieszczone w pojeździe transportowym, monitorujące temperaturę w regularnych odstępach czasu i wysyłających odczyty do aplikacji zarządzania łańcuchem dostaw.
- Centralna baza danych, która przechowuje dane zgodnie z logiką aplikacji zarządzania łańcuchem dostaw. Są to w szczególności dane pozyskane z czujników IoT, a także informacje o produktach, użytkownikach aplikacji itp.
- Oprogramowanie zarządzania łańcuchem dostaw, które jest odpowiedzialne za przetwarzanie danych otrzymywanych od czujników IoT, ich przechowywanie w centralnej bazie danych, analizę i prezentowanie użytkownikom końcowym.



Rysunek 21 Schemat realizacji systemu monitorowania łańcucha dostaw w wersji scentralizowanej

Zaproponowany schemat (Rysunek 21) wskazuje na kluczową rolę scentralizowanego oprogramowania do zarządzania łańcuchem dostaw. Krytyczne jest zapewnienie jego niezawodności, tak aby problemy z dostępnością usługi nie wpłynęły na możliwość odczytu danych napływających z czujników. Istotne jest także zagwarantowanie odpowiedniej wydajności tak, aby wykluczyć niedostępność usługi spowodowanej jej czasowym przeciążeniem, np. w wyniku ataku DoS lub DDoS [137]. Integralność danych musi być zapewniona podczas transmisji i przechowywania w centralnej bazie danych. Dostęp do danych

jest możliwy jedynie poprzez organizację posiadającą wdrożone rozwiązanie i jest ograniczony jedynie do danych tejże organizacji.



Rysunek 22 Schemat realizacji systemu monitorowania łańcucha dostaw w wersji zdecentralizowanej

Powyższy schemat (Rysunek 22) przedstawia sposób realizacji analizowanego przypadku użycia z wykorzystaniem technologii rejestru rozproszonego. Przewiduje on, że czujniki IoT będą mierzyły temperaturę w regularnych odstępach czasu i zapisywały ją bezpośrednio w rejestrze Hyperledger Fabric. Pozwoli to na zagwarantowanie niezmienność danych oraz ich rozproszenie w sieci HLF. Ze względu na występowanie wielu węzłów sieci, dla której każdy z interesariuszy może obsługiwać własną instancję lub instancje, zagwarantowana jest wysoka dostępność usługi. Zwiększenie wydajności może zostać osiągnięte poprzez skalowanie poziome [138], polegające na dodaniu dodatkowych węzłów do sieci, a tym samym zwiększenie jej możliwości składowania danych i obsługi ruchu.

Użytkownicy systemu zdecentralizowanego, będący jednocześnie uczestnikami łańcucha dostaw mogą uzyskać dostęp do przechowywanych danych bezpośrednio poprzez połączenie z siecią Hyperledger Fabric lub poprzez oprogramowanie zarządzania łańcuchem dostaw. Powyższe gwarantuje dostęp do składowanych danych, także w przypadku niedostępności oprogramowania wywołanej awarią lub przerwą konserwacyjną. Jednocześnie pozwala to uczestnikom łańcucha dostaw na porównanie danych składowanych w dostępnym dla nich rejestrze z danymi prezentowanymi przez aplikację. Każdy z uczestników może w czasie rzeczywistym śledzić warunki, w jakich przewożone są produkty oraz weryfikować

integralność tych danych. Łatwo zauważyć, iż uczestnikiem sieci może być np. organ nadzoru, który dzięki dostępowi do danych jest w stanie na bieżąco reagować na wszelkie naruszenia procedur dotyczących transportu produktów.

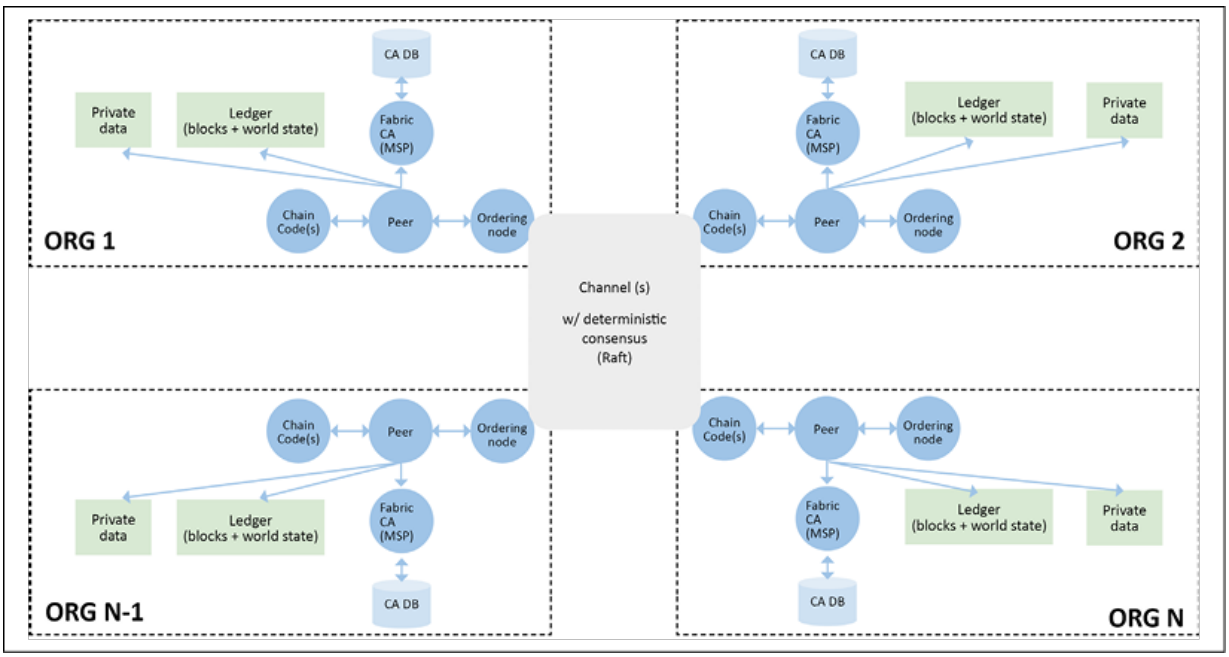
Oprogramowanie do zarządzania łańcuchem dostaw, w przypadku zastosowania podejścia zdecentralizowanego, nie jest punktem krytycznym dla realizacji zakładanego przypadku użycia. Jego głównym celem jest agregacja danych i ich prezentacja użytkownikom, podczas gdy główne funkcjonalności biznesowe mogą zostać zrealizowane przez Hyperledger Fabric. W szczególności istnieje możliwość wdrożenia smart kontraktów, które w sieci HLF są określane jako *chaincode* [139]. Kontrakt może wyzwolić akcję, np. wysłać określone powiadomienia do operatora, jeżeli zarejestrowany odczyt temperatury wyjdzie poza ustalony zakres.

Zaproponowany dla opisywanego przypadku użycia rejestr rozproszony – Hyperledger Fabric stanowi platformę prywatną, zaprojektowaną głównie do zastosowań korporacyjnych. Rekomendacja ta jest podyktowana oferowanym przez HLF zaawansowanym mechanizmem kontroli dostępu do danych oraz zachowania ich prywatności. W przypadku łańcucha dostaw różne podmioty, takie jak producenci, logistyka, hurtownicy, czy też organy nadzoru muszą mieć dostęp do zgromadzonych danych jedynie w takim zakresie, jaki jest niezbędny do ich pracy. Sieć HLF pozwala na tworzenie kanałów, w których transakcje są widoczne tylko dla autoryzowanych użytkowników, np. odczyty dla produktów transportowanych przez firmę *X* nie są widoczne dla innej firmy transportowej *Y*.

Blockchain Hyperledger Fabric został oparty o konsensus PBFT [26], który pozwala na przeprowadzanie transakcji bez wykonywania kosztownych obliczeń dla konsensusu Proof of Work, a także bez konieczności ponoszenia opłat transakcyjnych związanych z przechowywaniem danych w sieci publicznej. Ze względu na swój prywatny charakter, przeznaczony do zastosowań korporacyjnych, HLF nie wymaga wypłacania nagród uczestnikom sieci, aby zachęcić ich do utrzymywania aktywnych węzłów. Wykorzystana modularna architektura tego rejestru rozproszonego pozwala na łatwą skalowalność rozwiązania poprzez zwiększanie liczby węzłów sieci. Prywatny charakter HLF pozwala także na pełną przewidywalność działania oraz stabilność, która nie jest osiągalna przez rejestry publiczne [140].

Zastosowanie kontraktów w sieci Hyperledger Fabric umożliwia rozproszone wykonywanie kodu odpowiedzialnego za nadzór nad łańcuchem dostaw. W znaczący sposób

zwiększa to odporność systemu na awarie, czy też błędy ludzkie. Audytowalny kod kontraktu w sposób automatyczny realizuje założoną logikę biznesową. Informacje o wykrytych naruszeniach w łańcuchu dostaw są w sposób automatyczny dystrybuowane do zdefiniowanych w kontrakcie odbiorców, wykluczając możliwość manualnej interwencji w ten proces.



Rysunek 23 Architektura Hyperledger Fabric [141]

Zaproponowany schemat (Rysunek 22) wykorzystujący technologię DLT zapewnia szereg zalet w stosunku do schematu scentralizowanego (Rysunek 21). W tradycyjnym systemie, dane są zazwyczaj zarządzane przez jedną organizację lub platformę, co skutkuje powstaniem pojedynczego punktu awarii. Tradycyjne podejście może powodować problemy z bezpieczeństwem i prywatnością danych, szczególnie w przypadku występowania błędów implementacyjnych wprowadzonych do centralnego oprogramowania zarządzającego danymi. Wykorzystanie Hyperledger Fabric pozwala na rozproszenie danych i logiki biznesowej pomiędzy uczestnikami łańcucha dostaw. Każdy z uczestników może posiadać własny węzeł w sieci, co zwiększa jej decentralizację, a tym samym integralność zawartych w niej danych.

Dostęp do danych w sieci HLF jest kontrolowany przez uczestników sieci, jednakże każdy z aktorów ma niezależność w zarządzaniu swoimi danymi i transakcjami. Pomimo występowania podobieństwa do systemów scentralizowanych, podejście to pozostawia organizacjom swobodę w kontroli swoich danych zapewniając decentralizację zarządzania i operacji.

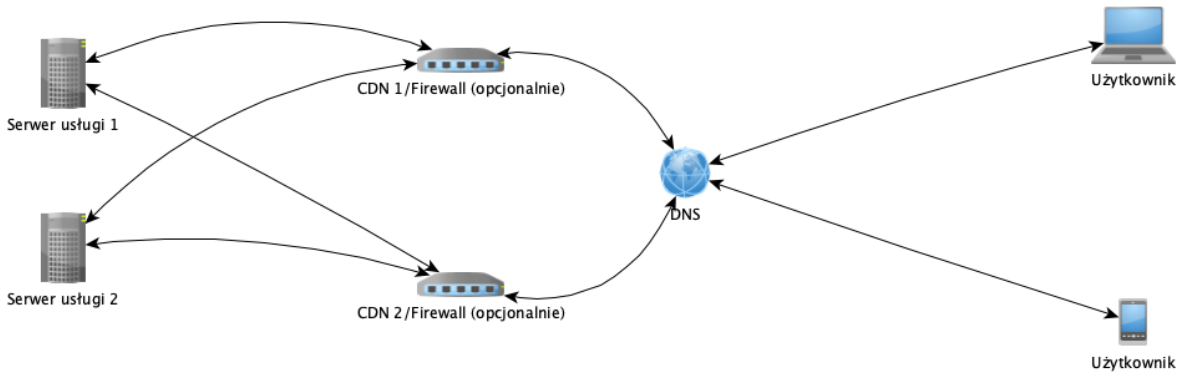
Zastosowanie prywatnej platformy blockchain do realizacji opisywanego przypadku użycia gwarantuje wysoki poziom integralności przetwarzanych danych, przy jednoczesnym zredukowaniu wymaganego poziomu zaufania do uczestników sieci. W stosunku do wykorzystania rejestru rozproszonego opartego o sieć publiczną, taką jak Ethereum, zaproponowane rozwiązanie pozwala na zwiększenie wydajności sieci, gwarantując przy tym poufność przechowywanych w nim danych. Uczestnicy sieci zyskują możliwość współdzielenia danych w sposób bezpieczny, transparentny i odporny na manipulację. Przy tym wszystkim zachowują kontrolę nad własnymi informacjami i procesami.

5.2. Dostępność

Niniejszy podrozdział **prezentuje autorskie koncepcje zwiększenia dostępności danych** w sieciach opartych o technologie rejestru rozproszonego, takie jak blockchain. Przeprowadzono w nim analizę metod decentralizacji infrastruktury poprzez replikację danych na liczne węzły sieci, w kontekście poprawy niezawodności i dostępności danych oraz uniemożliwienia cenzurowania treści. Przedstawione zostały także statystyki i przykłady ataków DDoS, które pokazują rosnące zagrożenie dla systemów scentralizowanych. Podrozdział wskazuje potencjalne zastosowania i zalety wykorzystania zdecentralizowanego przechowywania danych, szczególnie w kontekście zapewnienia dostępu do informacji w sytuacjach kryzysowych lub przy ograniczonej łączności.

Technologie rejestru rozproszonego w znakomitej większości opierają się na koncepcji tworzenia połączeń pomiędzy użytkownikami sieci – na zasadzie każdy-z-każdym (peer-to-peer, P2P) [142]. Zdecentralizowanie infrastruktury, poprzez replikację zawartości na wiele węzłów sieci, w znaczący sposób wpływa na zwiększenie niezawodności sieci oraz dostępności udostępnianych przez nią treści. Zastosowanie tej technologii pozwala także na wyeliminowanie możliwości cenzurowania treści, poprzez pominięcie pośredników w procesie jej transmisji. Powyższe pozwala na stworzenie odpornej infrastruktury, pozwalającej na wymianę danych pomiędzy uczestnikami sieci w warunkach ograniczonej łączności lub odcięcia dostępu do sieci zewnętrznej [143].

Zgodnie z [144] liczba ataków DDoS (Distributed Denial of Service) w ostatnim kwartale 2023 wynosiła 57116. To samo źródło wskazuje także na stały wzrost ataków tego typu – porównując rok 2021 do roku 2022, liczba ataków wzrosła o 111%. Średni czas trwania jednego ataku wzrósł z 30 do 50 minut. Jeden z najbardziej istotnych ataków, który był obsłużony przez Google Cloud Armor, w szczytowym momencie osiągnął 46 milionów zapytań na sekundę. Brało w nim udział ponad 5000 adresów IP ze 132 krajów. 30% ruchu sieciowego było generowane z krajów takich jak Brazylia, Indie, Rosja i Indonezja. Ten rodzaj ataków jest skutecznie wykorzystywany przy prowadzeniu działań w cyberprzestrzeni, nakierowanych na infrastrukturę krytyczną i zasoby przeciwnika. Ataki DoS i DDoS były szeroko wykorzystywane podczas wojny rosyjsko-ukraińskiej [145], powodując ograniczenia w dostępności usług i informacji dla obywateli obydwóch państw.



Rysunek 24 Schemat przepływu żądań dla scentralizowanej witryny internetowej

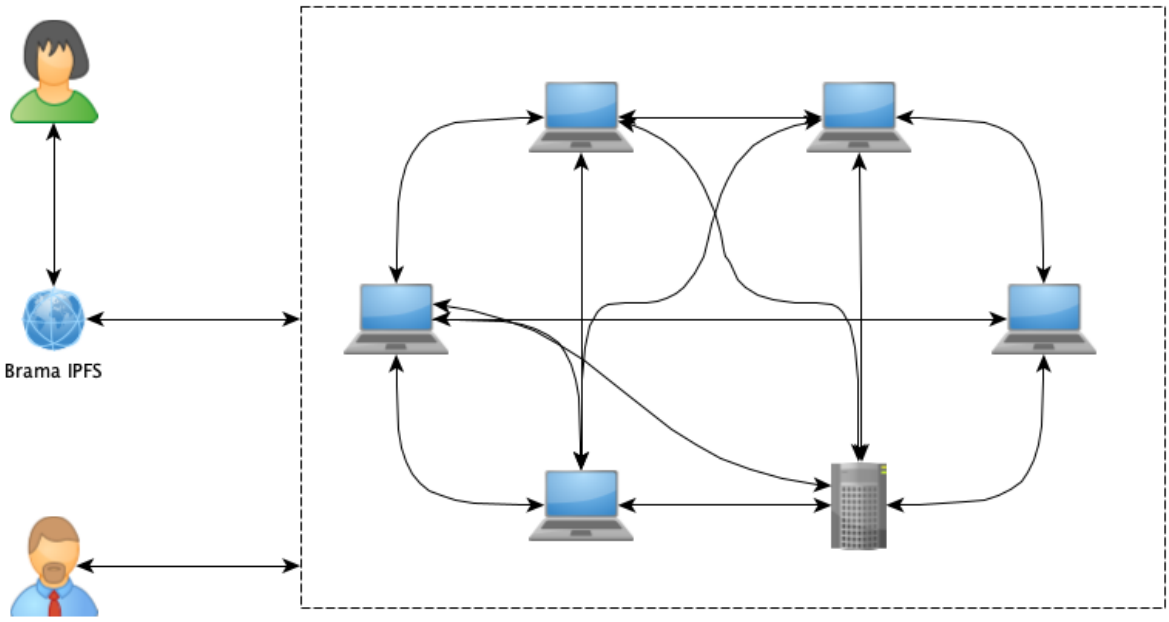
Ataki nakierowane na ograniczenie dostępności usługi mogą być adresowane z wykorzystaniem dedykowanych usług dokonujących filtrowania ruchu, detekcji niepożądanych wzorców, a także zwiększania możliwości obsługi ruchu przychodzącego [146]. Rozwiązania te w znaczący sposób zwiększają dostępność usług, jednak nie adresują problemów związanych z cenzurowaniem treści, czy też odcięciem danego terytorium od dostępu do globalnej sieci. Rozważmy schemat, w którym instytucja rządowa prowadzi witrynę internetową pozwalającą obywatelom na bieżący dostęp do przekazywanych komunikatów dotyczących aktualnej sytuacji w kraju. W tradycyjnym modelu zakładającym centralizację zasobów, strona taka jest hostowana z wykorzystaniem jednego lub wielu serwerów. Serwery te są pod kontrolą instytucji lub zewnętrznego dostawcy usług hostingowych. Użytkownicy, w celu uzyskania dostępu do zawartych na stronie treści, muszą połączyć się z serwerami przechowującymi odpowiednie dane. Podstawową wadą tego podejścia jest istnienie pojedynczych punktów awarii. Może być to awaria serwera przechowującego treści, jak i awarie urządzeń sieciowych przekazujących do niego ruch. W przypadku wystąpienia gwałtownego wzrostu obciążenia, podejście scentralizowane może nie zapewnić wystarczających zasobów serwerowych do obsługi napływających żądań. Proces zwiększania wydajności może okazać się kosztowny i czasochłonny, a w sytuacjach kryzysowych niemożliwy do zrealizowania. Rozwiązanie scentralizowane jest także podatne na cenzurę, w szczególności, jeżeli wykorzystywane są zasoby dostawcy zewnętrznego. W sytuacji wystąpienia kryzysu politycznego, dostawca może odmówić dalszego świadczenia usług, co będzie skutkowało brakiem ich dostępności.

Uproszczony schemat rozwiązania scentralizowanego (Rysunek 24), zakłada wykorzystanie DNS do rozwiązywania nazw domenowych przez użytkowników, opcjonalnie CDN (Content Delivery Network) do rozproszenia plików na serwerach dostarczających treści

oraz przechowywania najczęściej wykorzystywanych danych w pamięci podręcznej, Firewall którego celem jest filtrowanie ruchu sieciowego oraz serwery fizycznie hostujące dane. W przedstawionym schemacie można wyróżnić następujące punkty awarii:

- DNS – awaria DNS uniemożliwi rozwiązanie adresu usługi, powodując jej niedostępność. W przypadku przekierowania ruchu sieciowego istnieje zagrożenie naruszenia integralności danych.
- CDN – w przypadku awarii systemu CDN treści nie będą dostępne dla użytkowników końcowych. Ryzyko może być mitygowane przez wybór usługi o wysokiej dostępności.
- Firewall – w przypadku awarii lub celowej modyfikacji konfiguracji możliwe jest całkowite wyłączenie usługi.
- Serwer usługi – w zależności od sposobu działania systemu CDN, istnieje możliwość ograniczenia dostępności w przypadku awarii jednego z serwerów hostujących dane.

Rozważany przypadek użycia – witryny internetowej instytucji rządowej przeznaczonej do jednokierunkowej komunikacji z obywatelami, pozwala na zastosowanie decentralizacji infrastruktury w celu zwiększenia dostępności i zagwarantowania integralności danych. Rozwiązaniem technologicznym, które pozwala na realizację tego schematu, jest IPFS [147]. Wykorzystanie protokołu peer-to-peer (każdy z każdym) do przechowywania danych pozwala na eliminację pojedynczych punktów awarii. Pliki są rozproszone pomiędzy węzłami sieci, zatem awaria części z nich nie wpływa na dostępność tak hostowanej strony. Każdy z użytkowników, który pełni jedocześnie rolę węzła sieci IPFS, uzyskując dostęp do treści witryny pobiera jej zawartość. Następnie zawartość ta może być udostępniana z urządzenia użytkownika, co pozwala na automatyczne skalowanie sieci i dalsze zwiększanie dostępności bez ponoszenia kosztów dostosowania infrastruktury.

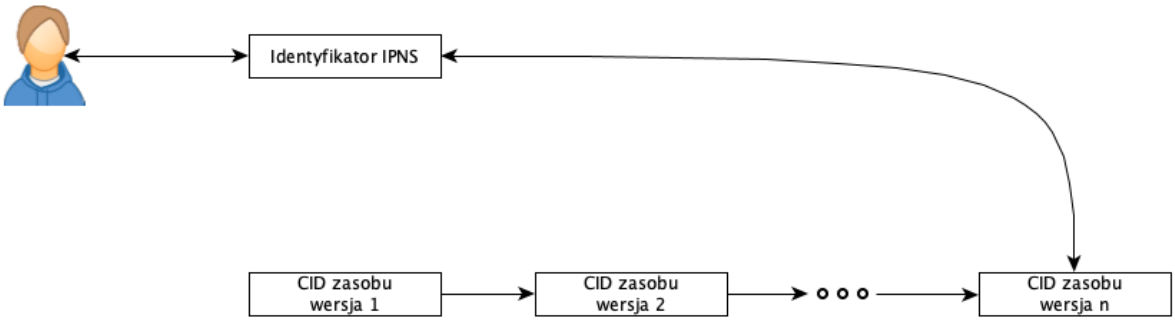


Rysunek 25 Schemat połączeń dla witryny internetowej hostowanej w IPFS

Decentralizacja w znaczący sposób utrudnia blokowanie dostępu do zawartości przechowywanej w sieci IPFS strony [148]. Brak jednej lokalizacji oznacza konieczność wyeliminowania wszystkich węzłów sieci przechowujących daną treść. Przechowywanie witryny na węzłach sieci IPFS przynosi także możliwość uzyskania dostępu do wcześniej pobranych treści, bez potrzeby posiadania aktywnego połączenia z pozostałymi węzłami. Pozwala to na realizację scenariusza dostępu do zawartości witryny w trybie offline.

Zaprezentowany na rysunku 25 schemat wskazuje dwa podstawowe sposoby pozyskania zawartości witryny internetowej. Pierwszym z nich jest wykorzystanie bramy IPFS – usługi, która pozwala na uzyskanie dostępu do sieci IPFS bez posiadania zainstalowanego klienta protokołu i utrzymywania jej pełnego węzła [149]. Brama stanowi alternatywę dla użytkowników, którzy nie chcą lub nie mogą utrzymywać pełnego węzła sieci. Ze względu na swoje umiejscowienie na styku sieci IPFS/HTTP i pełnioną rolę pośrednika w komunikacji, brama stanowi pojedynczy punkt awarii, a jej wykorzystywanie powinno być maksymalnie ograniczane. Organizacja utrzymująca bramę ma wpływ na transmitowane treści i może dokonać ich cenzury, w szczególności poprzez całkowite wykluczenie dostępu [150]. Drugim ze sposobów pozyskania treści jest bezpośrednie pobranie materiałów z sieci IPFS. W tym celu klient musi posiadać aktywne połączenie z siecią – skonfigurowane i aktywne oprogramowanie klienckie, a także znać adres CID poszukiwanego zasobu. CID może wskazywać na pojedynczy plik lub katalog zawierający pliki składające się na witrynę.

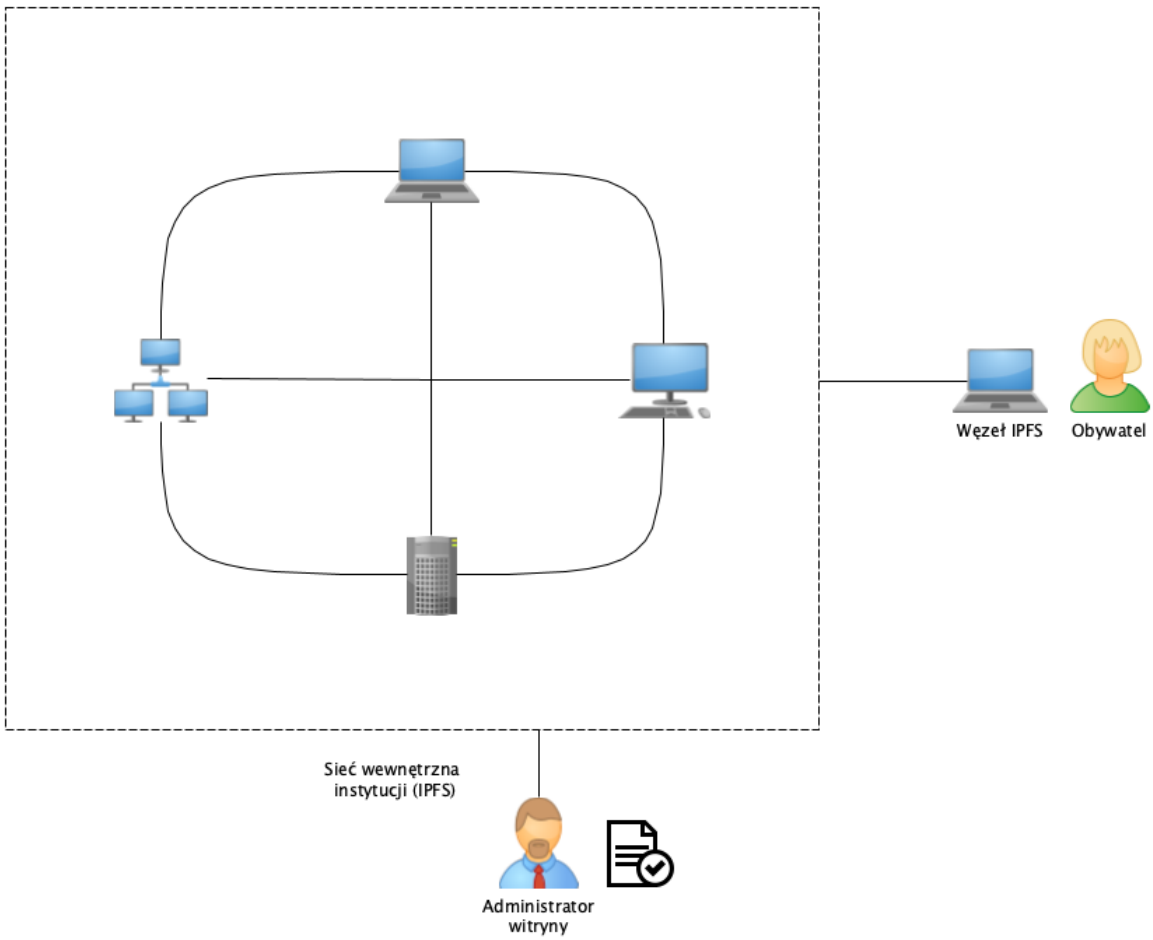
Sieć IPFS stosuje adresowanie zawartością, co oznacza, że CID jest wyznaczany na podstawie skrótu kryptograficznego danych. W przypadku wykorzystania IPFS do hostowania witryny internetowej implikuje to, iż po każdorazowej aktualizacji treści witryny, wartość CID zostanie zmieniona. Powyższe stanowi znaczące utrudnienie w dystrybucji treści, ponieważ wymaga utrzymywania niezależnego kanału, w którym publikowane są nowe wartości CID. Kanał taki musi być zaufany przez użytkowników, w przeciwnym przypadku istnieje ryzyko przekierowania do treści wskazanych przez zarządzający nim podmiot. Problem ten został zaadresowany przez twórców protokołu IPFS, w postaci implementacji rozwiązania IPNS (InterPlanetary Name System) [151]. IPNS pozwala na tworzenie trwałych i mutowalnych linków do zasobów umieszczonych w sieci IPFS.



Rysunek 26 Schemat działania IPNS

Rekord IPNS jest powiązany z parą kluczy kryptograficznych wykorzystywanych do wygenerowania identyfikatora IPNS oraz podpisywania zamieszczonych rekordów. Identyfikator ma postać skrótu kryptograficznego klucza publicznego właściciela rekordu, pozwalając na zapewnienie unikalności oraz możliwości weryfikacji autentyczności rekordu z wykorzystaniem odpowiadającego mu klucza. Właściciel właściwego klucza publicznego może opublikować nowy wpis zawierający CID do zasobu IPFS, jaki ma być wskazywany przez ten rekord. Wpis może zawierać informacje o czasie życia rekordu (TTL), który określa jak długo dany rekord może zostać uznawany za ważny. Użytkownik sieci, chcąc uzyskać dostęp do zasobu wskazanego przez IPNS kwerenduje sieć IPFS w celu odnalezienia odpowiadającego wpisu. Po jego odnalezieniu weryfikuje umieszczony podpis cyfrowy, który ma za zadanie zapewnić integralność i uwierzytelnienie autora rekordu. Jeżeli weryfikacja zakończy się powodzeniem, użytkownik otrzymuje identyfikator CID zasobu sieci IPFS i może uzyskać do niego dostęp. Możliwe jest także wykorzystanie istniejących nazw domenowych, poprzez użycie rekordów DNSLink [152]. Wpis ten w powiązaniu z IPNS pozwala na

wykorzystanie nazw domenowych przyjaznych dla użytkownika, przy jednoczesnym zapewnieniu zalet wynikających z IPNS.



Rysunek 27 Schemat udostępniania zasobów w prywatnej sieci IPFS

Opisany powyżej schemat opiera się na założeniu, że uczestnicy sieci korzystają z publicznych węzłów sieci IPFS oraz sieci dostępnej z wykorzystaniem połączenia internetowego. W sytuacji, gdy zapewnienie dostępności usług jest krytyczne, instytucja może zdecydować się na udostępnienie witryny w sieci niedostępnej z publicznego Internetu. Powyższe może mieć znaczenie w przypadku blokady dostępu do sieci Internet w skali kraju bądź regionu. Blokada takiego dostępu mogłaby wynikać np. z przyjętych środków zaradczych wobec działań nakierowanych na infrastrukturę teleinformatyczną kraju lub też być związana z blokadą danego kraju na węzłach stykowych sieci, wprowadzonych np. w formie sankcji. IPFS pozwala na zrealizowanie takiego przypadku użycia poprzez wykorzystanie sieci prywatnej, która składa się z serwerów i komputerów danej instytucji oraz urządzeń obywateli, tworząc lokalną sieć dystrybucji treści. Każdy z węzłów takiej sieci posiada zdolność przechowywania, żądania i dostarczania danych w ramach prywatnej sieci IPFS.

Realizacja schematu udostępniania treści w sieci wewnętrznej (Rysunek 27) zakłada wykorzystanie infrastruktury instytucji, pozwalając na dołączenie do niej urządzeń obywateli. Każdy dołączany węzeł powinien posiadać równe prawa w zakresie dostępu do danych, w szczególności powinien posiadać możliwość rozwiązywania adresów CID na podstawie rekordów IPNS zasobu udostępnianego przez instytucję. Instytucja udostępnia materiały z wykorzystaniem roli Administratora witryny, posiadającego zestaw kluczy kryptograficznych niezbędnych do zaktualizowania wpisu IPNS. Sieć może hostować wiele witryn z wykorzystaniem IPNS – jedna witryna wykorzystuje jedną parę kluczy, jednakże nie ma ograniczeń na ilość kluczy możliwych do używania przez dany węzeł sieci. Treści, które zostały pobrane z węzłów znajdujących się pod kontrolą instytucji będą dostępne na węzłach obywateli pozwalając na zwiększenie wydajności, dostępności i niezawodności sieci. Zastosowanie mechanizmów kryptograficznych do aktualizacji wpisów IPNS daje użytkownikom gwarancję pochodzenia treści, a także ich integralności.

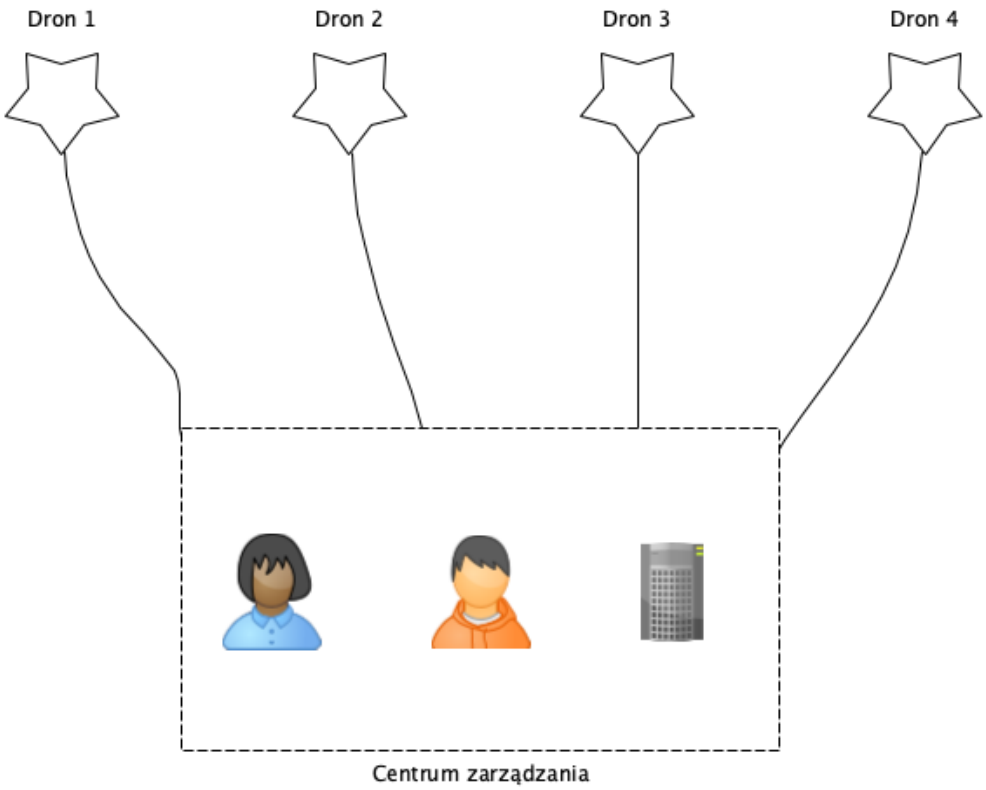
Studium powyższego przypadku wskazuje na znaczące zwiększenie dostępności usług przy wykorzystaniu technologii IPFS. Decentralizacja infrastruktury i związane z nią zwiększenie wydajności, skalowalności i ogólnej odporności systemu wskazuje ścieżkę rozwoju dla systemów, w których te cechy stanowią o realizacji zakładanego przypadku użycia. Ściśle zintegrowane z IPFS zapewnienie integralności danych stanowi doskonały przykład wykorzystania rejestru rozproszonego, w którym uczestnicy sieci mogą prześledzić zmiany dokonywane np. we wpisach IPNS, pozwalając na przechowywanie pełnej historii zamieszczanych informacji. Cecha ta zapewnia dostęp do informacji, które mogłyby zostać usunięte, np. w wyniku przejęcia kluczy kryptograficznych i modyfikacji wpisów IPNS przez atakującego.

5.3. Obliczenia rozproszone

Niniejszy podrozdział wprowadza w **autorskie rozważania na temat wykorzystania obliczeń rozproszonych w oparciu o technologie rejestru rozproszonego** (DLT). Omówiono, jak decentralizacja zasobów obliczeniowych i danych wpływa na przetwarzanie informacji w zdecentralizowanej sieci złożonej z serwerów, komputerów osobistych i urządzeń IoT. Szczególną uwagę skupiono na wykorzystaniu tej technologii w zarządzaniu autonomicznymi rojami dronów, co pozwala na większą odporność na zakłócenia i ataki radioelektroniczne, stanowiące wyzwanie w nowoczesnym polu walki. Proponowane autorskie podejście pozwala na zwiększenie elastyczności i dynamiki operacji, co jest kluczowe w szybko zmieniających się warunkach operacyjnych.

Obliczenia rozproszone [153] wykorzystujące technologię rejestru rozproszonego to podejście, które łączy w sobie rozproszenie zasobów obliczeniowych i przechowywanie danych w sieci zdecentralizowanej. W tym modelu dane są przetwarzane przez wiele węzłów, które działają niezależnie, współpracując w celu osiągnięcia wspólnego celu. W odróżnieniu od tradycyjnych systemów scentralizowanych, obliczenia rozproszone wykorzystujące DLT wykorzystują zdecentralizowaną sieć węzłów takich jak serwery, komputery osobiste czy urządzenia IoT. Każdy z nich może przyczyniać się do procesu decyzyjnego, przetwarzania danych lub weryfikacji transakcji.

Problem kooperacji urządzeń, prowadzenia ich odpornej komunikacji oraz wspólnych obliczeń został poruszony w [154]. Powyższe stanowi szczególne wyzwanie w przypadku urządzeń bezzałogowych, których zadaniem jest operowanie na nowoczesnym polu walki [155]. Urządzenia działające w terenie przyfrontowym, czy też bezpośrednio w regionie walk są narażone na działanie systemów walki radioelektronicznej (WRE) [156], zdolnych do ograniczenia możliwości komunikacyjnych, zakłócenia odczytu z czujników pokładowych, czy też całkowitego uszkodzenia urządzenia z wykorzystaniem impulsów elektromagnetycznych. Rozważmy następujący przypadek użycia – jednostka wojskowa dysponuje rojem dronów wykorzystywanych do ataku na wskazane cele.



Rysunek 28 Schemat scentralizowanego zarządzania rojem dronów

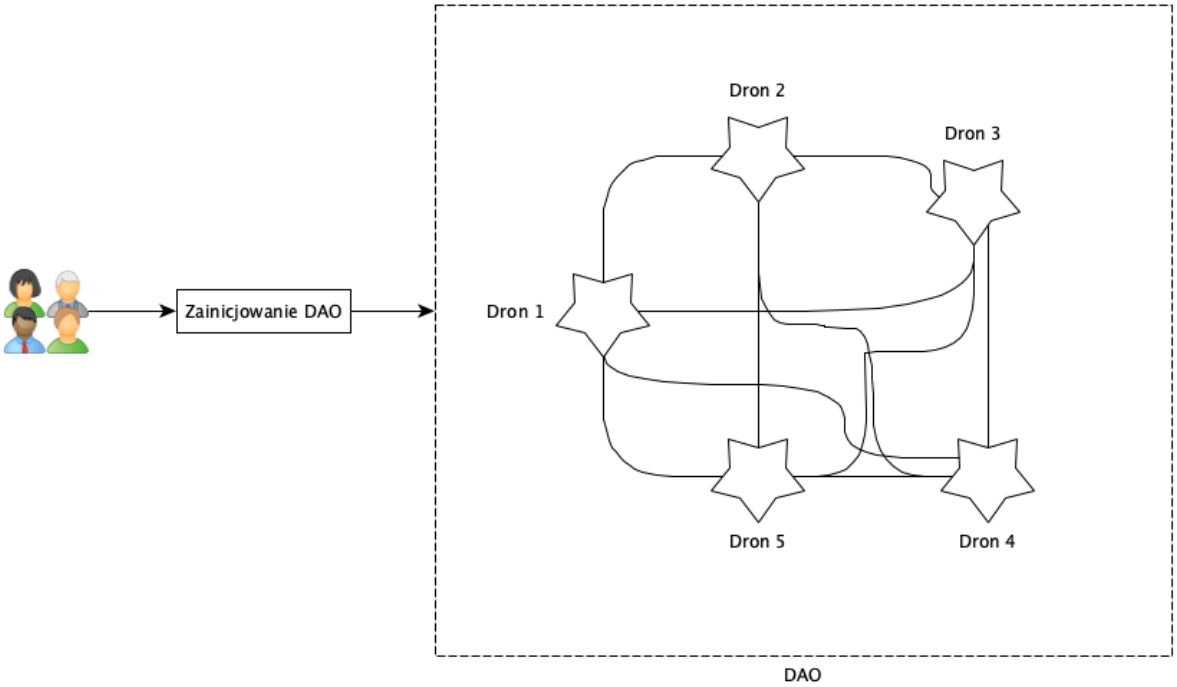
W tradycyjnym, scentralizowanym podejściu (Rysunek 28) drony te są sterowane z centralnego punktu zarządzania. Może to być zarówno stacjonarny budynek zlokalizowany na tyłach lub stacja mobilna zlokalizowana w bliskiej odległości od strefy działania roju. Wszystkie decyzje operacyjne, nawigacja, identyfikacja celów oraz podejmowanie działań są przekazywane przez operatora ludzkiego lub centralny system komputerowy [157]. W tej konfiguracji istnieje ryzyko zwiększonej podatności na środki walki radioelektronicznej. W szczególności mogą to być:

- Próby zagłuszania lub zakłócania sygnałów komunikacyjnych pomiędzy dronami i centralnym punktem zarządzania.
- Podszywanie się pod sygnały nawigacyjne (np. GPS) lub komunikacyjne w celu wprowadzenia w błąd systemu nawigacyjnego urządzenia bezzałogowego.

Powyższe ryzyka są redukowane poprzez zastosowanie rozwiązań technologicznych takich jak szyfrowanie komunikacji, przeskakiwanie częstotliwości czy też wprowadzenie redundantnych metod łączności. Te środki zaradcze często okazują się niewystarczające, czego dobrym przykładem jest wykorzystanie WRE w trwającym konflikcie rosyjsko-ukraińskim [158].

Alternatywnym podejściem, pozwalającym na realizację autonomicznego roju dronów wykorzystującego zdecentralizowany system zarządzania, jest wykorzystanie technologii rejestru rozproszonego i zdecentralizowanego przechowywania danych. W podejściu tym drony tworzą DAO (zdecentralizowaną autonomiczną organizację), wspólnie ustalając reguły atakowania celu oraz wymieniając dane z czujników należących do członków sieci w celu podjęcia autonomicznej decyzji. Drony nie muszą otrzymywać bezpośredniej łączności z centralnym punktem zarządzania, wystarczającym jest zdefiniowania reguł przed rozpoczęciem danej misji.

Wykorzystanie DAO do zaimplementowania autonomicznego roju dronów przedstawia szereg korzyści w stosunku do rozwiązania scentralizowanego. Brak centralnego punktu, który mógłby podlegać atakom z wykorzystaniem środków WRE zwiększa odporność na zakłócenia i utratę łączności. Każdy z dronów stanowi niezależny byt, który stanowi część organizacji podejmującej działania zgodnie z określonymi regułami. Proces podjęcia decyzji, np. o ataku, zmianie trasy podejścia, zaniechaniu akcji jest wykonywany na podstawie z góry ustalonych reguł i algorytmów. W procesie decyzyjnym biorą udział wszyscy uczestnicy sieci, dostarczając jednocześnie informacji zebranych z wykorzystaniem własnych czujników. W przypadku roju dronów mogą być to informacje o wykrytych obiektach, zmianie warunków otoczenia, nadchodzącym ataku na rój itp. Skrócenie ścieżki decyzyjnej oraz zmniejszenie związanego z nią opóźnienia pozwala na większą elastyczność i dynamiczną adaptację do zmieniających się warunków, w których operuje rój. Przykładowo – rój dronów podczas natarcia na zbliżającą się kolumnę pancerną może identyfikować sprzęt, którego zniszczenie przyniesie maksymalne korzyści. DAO może zmienić pierwotną decyzję dotyczącą atakowania jedynie wybranego typu pojazdów, adaptując ją do zastanych warunków pola bitwy. Rój może zmienić zdefiniowany inicjalnie cel – z atakowania jedynie czołgów podstawowych do atakowania osłaniających natarcie śmigłowców szturmowych, czy też podążających za kolumną komponentów systemu obrony przeciwlotniczej. System taki w obliczu braku scentralizowanej formy komunikacji i zarządzania jest zdecydowanie trudniejszy do wyeliminowania, ponieważ wymaga unieszkodliwienia wszystkich członków DAO [159].



Rysunek 29 Schemat roju dronów w formie DAO

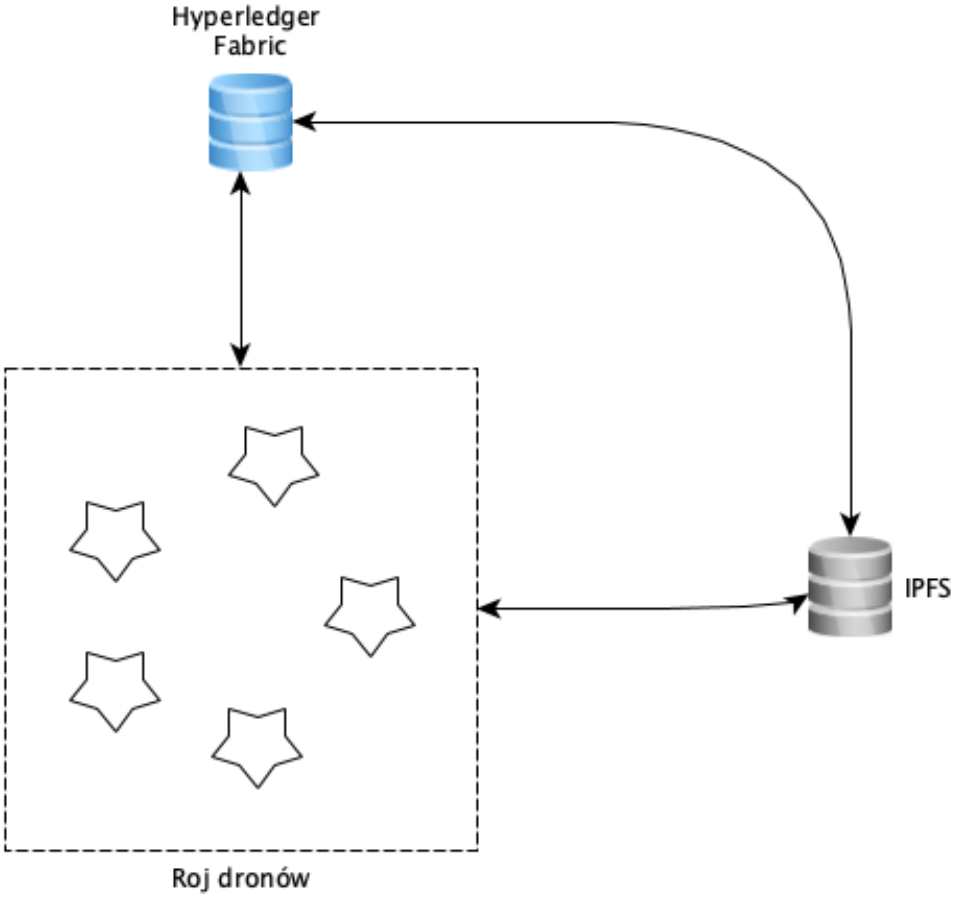
Podstawowymi wymaganiami projektowymi dla DAO, które może pozwolić na realizację **autonomicznego roju dronów jest zapewnienie niezawodnej komunikacji, skutecznego procesu decyzyjnego i odporności na zakłócenia**. Zastosowanie technologii blockchain oraz sieci mesh [160] do komunikacji pozwala na zbudowanie zdecentralizowanego i odpornego systemu. Poniżej przedstawiony został ogólny zarys takiego systemu:

- Każdy dron jest rejestrowany w sieci z unikalnym identyfikatorem i parą kluczy kryptograficznych. DAO rejestruje drony z wykorzystaniem identyfikatora i klucza publicznego przechowywanego w rejestrze blockchain.
- W zależności od rodzaju pełnionych misji i specyfikacji dronów, mogą one otrzymać przypisanie roli w DAO. Przykładem takich ról może być zwiadowca lub dron atakujący.
- Rój komunikuje się z wykorzystaniem sieci mesh, pozwalającej na zdecentralizowaną komunikację bez konieczności utrzymywania stałego połączenia z zewnętrznym punktem kontrolnym.
- Protokoły komunikacyjne wykorzystywane przez rój są zaprojektowane do pracy w warunkach ograniczonej łączności, w szczególności tolerując opóźnienia i przerwy w komunikacji.

- Algorytm konsensusu wykorzystywany w sieci powinien być dostosowany do ograniczeń operacyjnych dronów, w szczególności do podejmowania decyzji na podstawie dostępnych danych sensorycznych i innych informacji pozyskanych od pozostałych członków DAO.
- Warunki misji muszą być zdefiniowane w smart kontraktach. Warunkiem może być np. przeprowadzenie ataku jedynie poprzez potwierdzenie celu przez określoną liczbę dronów.
- DAO powinno pozwalać na dynamiczną rekonfigurację, w przypadku zmiany warunków otoczenia. Przykładem może być zmiana celu ataku w przypadku wykrycia nowego zagrożenia lub utraty celu pierwotnego.
- Sieć musi zapewniać szyfrowanie przesyłanych komunikatów oraz eliminować możliwość dołączenia nieautoryzowanych klientów do DAO, a w szczególności do uczestnictwa w procesie decyzyjnym.

Praktyczna realizacja DAO może zostać wykonana z wykorzystaniem IPFS i Hyperledger Fabric [161]. Rolą Hyperledger Fabric jest implementacja smart kontraktów oraz mechanizmów konsensusu dla roju dronów. Ze względu na oferowaną szybkość transakcji, skalowalność oraz bezpieczeństwo wynikające z zastosowanych mechanizmów kontroli dostępu i poufności, HLF stanowi doskonałą bazę do implementacji systemu. Rolą IPFS jest zapewnienie wewnętrznej przestrzeni udostępniania danych pomiędzy członkami roju. Prywatna sieć P2P pozwala na współdzielenie pozyskanych informacji zwiadowczych pomiędzy dronami, bez konieczności obciążania wykorzystywanej sieci blockchain. Jest to szczególnie istotne w sytuacji, gdy nie wszyscy członkowie sieci HLF muszą mieć dostęp do surowych danych, na podstawie których podejmowane są decyzje.

Hyperledger Fabric zakłada elastyczność w doborze algorytmu konsensusu, dostarczając dwa podstawowe algorytmy, które mogą być wykorzystane przy implementacji DAO dla roju dronów. Należą do nich Raft [162] oraz Kafka [163]. Raft zakłada podejmowanie decyzji przez zdefiniowaną większość węzłów. Kafka była wykorzystywana głównie w Hyperledger Fabric 1 i jest aktualnie uznawana za algorytm przestarzały. Zastosowanie Raft do realizacji analizowanego w tym podrozdziale przypadku użycia pozwala na dostosowanie procesu decyzyjnego do zmieniającego się składu roju dronów. W przypadku utraty części z maszyn, możliwe jest zdefiniowanie nowego zestawu replikującego tak, aby uwzględnić zmienioną liczbę węzłów. W przypadku, gdyby uszkodzeniu uległ lider, algorytm ma możliwość samoczynnej elekcji nowego lidera spośród pozostałych węzłów.



Rysunek 30 Schemat roju wykorzystującego HLF i IPFS

Alternatywną wobec Hyperledger Fabric technologią, która mogłaby znaleźć zastosowanie w zarządzaniu rojem dronów jest IOTA. Nie wymaga ponoszenia opłat transakcyjnych, a ze względu na swoją konstrukcję oferuje dużą skalowalność, co jest szczególnie istotne przy zarządzaniu dużą liczbą urządzeń IoT, takich jak drony. Ze względu na swój otwarty charakter nie zapewnia realizacji skomplikowanych wymagań dotyczących prywatności i kontroli dostępu, dlatego jej wykorzystanie musi być poprzedzone odpowiednią analizą wymagań. W poniższej tabeli przedstawione zostało porównanie cech Hyperledger Fabric oraz IOTA.

Tabela 4 Porównanie cech Hyperledger Fabric i IOTA w kontekście zarządzania rojem dronów

Kryterium	IOTA	Hyperledger Fabric
Skalowalność	Bardzo wysoka, dzięki zastosowaniu struktury splotu pozwalającej na równoległe przetwarzanie transakcji.	Średnia, zależna od konfiguracji sieci i wybranego algorytmu konsensusu.

Opłaty za transakcje	Brak	Brak
Odporność na uszkodzenia węzłów	Wysoka. Każde urządzenie jest węzłem.	Średnia do wysokiej, zależnie od konfiguracji sieci. Mechanizmy konsensusu mogą się adaptować do zmiany liczby węzłów
Konsensus	Każda transakcje jest weryfikowana przez dwie inne. Brak klasycznego algorytmu konsensusu.	Konfigurowalny algorytm konsensusu, dostosowany do potrzeb aplikacji.
Przepustowość transakcji	Wysoka, wzrastająca wraz ze wzrostem sieci.	Zależna od konfiguracji sieci, wysoka w kontrolowanych i prywatnych środowiskach.
Prywatność i poufność	Ograniczona – wszystkie transakcje są widoczne w sieci, transakcje mogą być anonimowe, brak zaawansowanej kontroli dostępu.	Wysoka, z możliwością tworzenia prywatnych kanałów i konfigurowalnych ról użytkowników.
Bezpieczeństwo	Zależne od rozmiaru sieci, wzrasta wraz ze wzrostem liczby węzłów.	Wysokie, wykorzystujące zaawansowane techniki kryptograficzne.
Adaptacja do dynamicznych środowisk	Przystosowane do środowisk IoT cechujących się wysoką zmiennością.	Wymaga dodatkowej konfiguracji i zarządzania w szybko zmieniających się środowiskach.
Implementacja i utrzymanie	Łatwiejsze dla projektów IoT ze względu na prostotę modelu.	Złożona ze względu na skomplikowany proces konfiguracji sieci i implementacji smart kontraktów.

Wybór pomiędzy IOTA, a Hyperledger Fabric powinien zostać podjęty na etapie projektowania konkretnego rozwiązania, po szczegółowej analizie wymagań projektu. W szczególności istotne są aspekty zapewnienia prywatności, bezpieczeństwa i skalowalności.

Podsumowując, przedstawione w tym podrozdziale autorskie rozwiązania dotyczące obliczeń rozproszonych w oparciu o technologie DLT wskazują na ich znaczący potencjał w zwiększaniu odporności i niezawodności systemów operacyjnych. Szczególnie wartościowe okazują się w kontekście zarządzania rojem dronów, gdzie decentralizacja zarządzania i przetwarzania danych zapewnia wysoką odporność na zakłócenia zewnętrzne, takie jak ataki radioelektroniczne czy próby zagłuszenia sygnałów komunikacyjnych. Przyjęcie modelu DAO

do zarządzania rojem dronów **jest innowacyjnym podejściem, które znacznie zwiększa skuteczność działania w dynamicznych i wymagających środowiskach. Wdrożenie tych autorskich metod może prowadzić do rewolucji w sposobie myślenia o bezzałogowych systemach operacyjnych i otwiera drogę dla dalszych badań nad praktycznym zastosowaniem rozproszonych systemów obliczeniowych**.

6. Schemat uwierzytelnionego dostępu w środowisku rozproszonym

Rozdział ten prezentuje autorskie rozwiązania schematu uwierzytelnionego dostępu do danych w środowisku rozproszonym, korzystając z nowoczesnych technologii takich jak sieć IPFS, Ethereum oraz Secret Network. W pierwszym podrozdziale omówiono metody przechowywania tajnych kluczy w sieci blockchain, wykorzystując protokół Secret Network. Następnie, w podrozdziale dotyczącym autoryzacji, przedstawiono implementacje smart kontraktów i aplikacji webowych, które regulują dostęp do zasobów w zdecentralizowanym środowisku. Ostatni podrozdział koncentruje się na ocenie bezpieczeństwa i funkcjonalności zaproponowanego schematu, weryfikując spełnienie założonych wymagań funkcjonalnych i niefunkcjonalnych. Rozdział ten oferuje kompleksowe spojrzenie na wykorzystanie technologii blockchain do zaawansowanego zarządzania uwierzytelnieniem i autoryzacją w bezpiecznych systemach rozproszonych.

6.1. Przechowywanie sekretów w sieci blockchain

Niniejszy podrozdział skupia się na technologiach przechowywania sekretów w sieci blockchain, szczególnie z wykorzystaniem Secret Network, która wzbogaca standardową funkcjonalność blockchain o możliwości związane z prywatnością i anonimowością. Przedstawiono, w jaki sposób technologie dowodów o wiedzy zerowej i zaufane środowiska wykonawcze (TEE) mogą zabezpieczać smart kontrakty przed nieautoryzowanym dostępem, pozwalając jednocześnie na przetwarzanie wrażliwych danych. Omówiono architekturę i możliwości zastosowania sieci Secret Network. Podkreślono rozwiązania implementacji, które umożliwiają integrację z innymi sieciami opartymi o Cosmos SDK oraz szczegółowo opisano techniczne aspekty przechowywania i szyfrowania danych w kontraktach.

Sieci blockchain, takie jak Bitcoin, czy Ethereum zakładają pełną weryfikowalność zamieszczonych w nim danych przez użytkowników tych sieci. Oznacza to, że wszystkie dane, które są umieszczone w sieci, są publicznie dostępne, a kod kontraktów jest wykonywany w sposób deterministyczny. Cechy te należy uznać za wyjątkowo istotne w kontekście budowy systemu opierającego się o jawność, a nie zaufanie do wybranych jego uczestników. Przyjęte założenia powodują, że nie jest możliwe zrealizowanie niektórych przypadków użycia, w których prywatność użytkowników, czy ochrona wrażliwych danych stanowią istotę rozwiązania.

Problem zapewnienia anonimowości transakcji został zaadresowany poprzez wdrożenie rozwiązań opartych o dowody o wiedzy zerowej [164]. Zapewnienie prywatności i poufności dla smart kontraktów zostało zrealizowane dzięki połączeniu dowodów o wiedzy zerowej z wykonywaniem obliczeń w bezpiecznych enklawach sprzętowych [165]. Sieć blockchain, która zaimplementowała powyższe podejście, pozwalając na realizację złożonych smart kontraktów to Secret Network [166]. Rozwiązanie to zostało oparte o Cosmos SDK i wykorzystuje algorytm konsensusu typu Proof of Stake. Prywatność przetwarzanych danych jest gwarantowana poprzez wykorzystanie węzłów sieci posiadających zaufane środowisko uruchomieniowe (TEE, Trusted Execution Environment) zgodne ze specyfikacją sieci. Celem zastosowania TEE jest zagwarantowanie braku dostępu do wyników prowadzonych obliczeń przez węzły sieci.

Zaufane środowisko obliczeniowe tworzy w sieci Secret Network neutralnego uczestnika, który poprzez swoją formę dedykowanego rozwiązania sprzętowego, pozwala na prowadzenie bezpiecznych i prywatnych obliczeń. Powyższe może zostać porównane do roli pełnionej przez smart kontrakty w sieci Ethereum, które w sposób neutralny realizują zapisaną w kontrakcie logikę. TEE jest zlokalizowane w izolowanej części procesora i z punktu widzenia systemu operacyjnego stanowi niezależne urządzenie [167]. Zadaniem TEE jest stworzenie zaufanego środowiska wykonawczego, w którym dane mogą być przechowywane i przetwarzane bez ryzyka naruszenia ich integralności i poufności. W aktualnej implementacji sieci Secret Network, rolę TEE pełni rozwiązanie Intel SGX [168]. Konstrukcja sieci zakłada możliwość zmiany wykorzystanego rozwiązania sprzętowego w przyszłych wersjach protokołu.

Kontrakty w sieci Secret Network są wdrażane z wykorzystaniem CosmWasm [169], co wynika bezpośrednio z zastosowania Cosmos SDK w procesie projektowania sieci. Wykorzystanie CosmWasm pozwala na integrację tak stworzonych kontraktów z innymi sieciami opartymi o Cosmos SDK. Do tego celu wykorzystywany jest protokół Cosmos IBC (Inter-Blockchain Communication) [170]. Kod CosmWasm powstaje poprzez kompilację kodu kontraktu napisanego w języku Rust. Twórcy argumentują wybór języka Rust, jako podyktowany sposobem zarządzania dostępem do pamięci oraz szerokim wsparciem ze strony społeczności. Każdy kontrakt pozwala na wykorzystanie szyfrowanego wejścia, stanu i wyjścia [171]. Następujące wejścia do kontraktu są przekazywane w postaci jawnej:

- wysokość bloku,
- czas,
- identyfikator łańcucha,

- nadawca,
- adres,
- wysłane środki,
- skrót kryptograficzny kontraktu.

Szyfrowaniu podlegają wiadomości przekazywane przez klientów sieci. Stan kontraktu, przechowywany w postaci wewnętrznej bazy danych, jest zawsze szyfrowany i znany kontraktowi jedynie wewnątrz TEE. Wyjścia kontraktu, które są zaszyfrowane, mogą być odczytane jedynie przez nadawcę transakcji oraz kontrakt.

Sieć Cosmos nie przewiduje możliwości weryfikacji tożsamości strony wysyłającej zapytania. W przypadku sieci Secret Network problem ten jest rozwiązany poprzez przypisanie do kontraktu zaszyfrowanego klucza, wykorzystywanego do weryfikacji tożsamości strony przesyłającej zapytanie. Wyjściem zapytania może być odwołanie do wywołania innego kontraktu, inicjalizacja kontraktu, instrukcje przesłania środków z salda kontraktu, a także sekcja danych w postaci bajtowej, możliwa do interpretacji przez klienta lub aplikację komunikującą się z kontraktem.

Kod kontraktu sieci Secret Network może w ramach danej transakcji wykonać następujące operacje bazodanowe w ramach danego TEE:

- `write_db(nazwa_pola, wartość);`
- `read_db(nazwa_pola);`
- `remove_db(nazwa_pola).`

Poniżej przedstawiony został listing kodu źródłowego dla operacji *write_db* [172]:

```
encryption_key = hkdf({
  salt: hkfd_salt,
  ikm: concat(consensus_state_ikm, field_name, contract_key),
});

encrypted_field_name = aes_128_siv_encrypt({
  key: encryption_key,
  data: field_name,
});

current_state_ciphertext = internal_read_db(encrypted_field_name);

if (current_state_ciphertext == null) {
  // field_name doesn't yet initialized in state
  ad = sha256(encrypted_field_name);
} else {
  // read previous_ad, verify it, calculate new ad
  previous_ad = current_state_ciphertext.slice(0, 32); // first 32 bytes/256
bits
```

```
  current_state_ciphertext = current_state_ciphertext.slice(32); // skip
first 32 bytes

  aes_128_siv_decrypt({
    key: encryption_key,
    data: current_state_ciphertext,
    ad: previous_ad,
  }); // just to authenticate previous_ad
  ad = sha256(previous_ad);
}

new_state_ciphertext = aes_128_siv_encrypt({
  key: encryption_key,
  data: value,
  ad: ad,
});

new_state = concat(ad, new_state_ciphertext);

internal_write_db(encrypted_field_name, new_state);
```

Listing kodu źródłowego dla operacji *read_db* [172]*:*

```
encryption_key = hkdf({
  salt: hkfd_salt,
  ikm: concat(consensus_state_ikm, field_name, contract_key),
});

encrypted_field_name = aes_128_siv_encrypt({
  key: encryption_key,
  data: field_name,
});

current_state_ciphertext = internal_read_db(encrypted_field_name);

if (current_state_ciphertext == null) {
  // field_name doesn't yet initialized in state
  return null;
}

// read ad, verify it
ad = current_state_ciphertext.slice(0, 32); // first 32 bytes/256 bits
current_state_ciphertext = current_state_ciphertext.slice(32); // skip first
32 bytes
current_state_plaintext = aes_128_siv_decrypt({
  key: encryption_key,
  data: current_state_ciphertext,
  ad: ad,
});

return current_state_plaintext;
```

Kod operacji *remove_db* jest następujący [172]:

```
encryption_key = hkdf({
  salt: hkfd_salt,
  ikm: concat(consensus_state_ikm, field_name, contract_key),
});

encrypted_field_name = aes_128_siv_encrypt({
```

```
  key: encryption_key,
  data: field_name,
});

internal_remove_db(encrypted_field_name);
```

Stan kontraktu jest przechowywany w sieci w postaci par klucz-wartość. W związku z powyższym, *nazwa_pola* wykorzystywana w kontrakcie musi pozostać stała pomiędzy jego kolejnymi wywołaniami. Klucz szyfrujący wykorzystywany w procesie wywołania funkcji jest generowany z wykorzystaniem HKDF-SHA256 [173]. W procesie generacji wykorzystywany jest IKM (Initial Keying Material) stanu konsensusu, nazwy pola oraz klucza kontraktu. Protokół przewiduje możliwość dodania dodatkowych danych, które pozwolą na wygenerowanie różnych kluczy, przy zapisie tych samych danych w różnym czasie. Klucz kontraktu jest unikalny dla każdego wdrożonego w sieci Secret Network kontraktu. Jego niepodrabialność ma na celu zagwarantowanie, że żaden złośliwy walidator sieci nie będzie w stanie zaszyfrować transakcji z wykorzystaniem swojego klucza szyfrującego i następnie odzyskać stanu kontraktu, wykorzystując klucz podrobiony. Dodatkowo, każdy kontrakt musi być unikalny, co oznacza, że wdrożenie dwóch kontraktów o takim samym kodzie nie spowoduje powiązania ze sobą ich stanów.

Jeżeli kontrakt jest inicjalizowany przez węzeł walidujący sieci, to wewnątrz TEE węzła klucz kontraktu jest generowany z wykorzystaniem HMAC-SHA256 identyfikatora podpisującego oraz klucza uwierzytelniającego. Klucz uwierzytelniający jest generowany z wykorzystaniem HKDF przy wykorzystaniu wartości soli oraz IKM zawierającego konkatenację IKM stanu konsensusu oraz identyfikatora podpisującego. Przy każdym wykonaniu kodu kontraktu, klucz kontraktu jest wysyłany do węzłów walidujących sieci. Po otrzymaniu klucza kontraktu przeprowadzana jest weryfikacja jego zgodności z oczekiwaną wartością.

Bezpieczeństwo protokołu sieci Secret Network zostało przez twórców zbudowane w oparciu o następujące założenia:

1. Każdy węzeł sieci jest niezaufany i jest utrzymywany przez złośliwego uczestnika.
2. Każdy węzeł sieci posiada bezpieczną enklawę, w postaci Intel SGX, która może wykonywać kod i przetwarzać dane w sposób zaufany. Oznacza to, że dane te nie mogą być odczytywane ani modyfikowane przez uczestnika hostującego węzeł.
3. Prymitywy kryptograficzne wykorzystane w protokole są bezpieczne.
4. Sieć posiada wspólny konsensus.

Na bazie powyższych założeń, twórcy zidentyfikowali następujące podatności, które zgodnie z ich deklaracjami mają charakter teoretyczny [174]:

- Deanonimizacja transakcji poprzez analizę długości szyfrogramów.
- Występowanie dwóch kontraktów o takich samych kluczach, skutkujące nieautoryzowanym dostępem do stanu kontraktu.
- Atak powtórzeniowy dla transakcji.
- Przywrócenie poprzedniego stanu bazy danych podczas wykonywania kodu kontraktu.
- Wyciek informacji związany z nieprawidłową implementacją funkcji w kontrakcie.
- Podatność TEE skutkująca ujawnieniem danych przechowywanych w enklawie.

Wskazane właściwości sieci Secret Network, wskazują na możliwość jej wykorzystania do przechowywania sekretów w publicznym blockchain, a także wykorzystania ich do realizacji zaawansowanych przypadków użycia. W kontekście realizacji schematu uwierzytelnionego dostępu w środowisku rozproszonym, Secret Network pozwala na realizację pośrednika reszyfrowującego dane (reencryption proxy [175]) udostępniane w środowisku sieci publicznej do upoważnionych użytkowników. Szczegółowe wykorzystanie kontraktów sieci zostało omówione w kolejnym podrozdziale.

6.2. Autoryzacja z wykorzystaniem smart kontraktów

Podrozdział ten omawia autorskie implementacje mechanizmów autoryzacji z wykorzystaniem smart kontraktów w środowisku rozproszonym, skoncentrowanych na zapewnieniu bezpieczeństwa i prywatności danych przechowywanych w publicznych sieciach blockchain. Zaprezentowane zostało kompleksowe podejście do przechowywania, szyfrowania i autoryzacji dostępu do danych za pomocą zdecentralizowanych aplikacji i smart kontraktów. Opisano szczegółowo proces tworzenia, umieszczania i odczytu szyfrowanych plików, z zastosowaniem technologii IPFS oraz Ethereum. Dodatkowo, przedstawiono autorski system autoryzacji w sieci Secret Network, który umożliwia przeszyfrowywanie danych dla autoryzowanych użytkowników. System ten opiera się na nowoczesnych rozwiązaniach kryptograficznych i wykorzystuje zdecentralizowane aplikacje do zarządzania dostępem do danych, co znacząco wpływa na bezpieczeństwo i prywatność w rozproszonych środowiskach informatycznych.

Realizacja uwierzytelnionego dostępu do materiałów przechowywanych w publicznej sieci jest zadaniem wymagającym zastosowania szeregu technologii pozwalających na obsługę procesu składowania prywatnych danych. W szczególności wymagane jest zastosowanie technologii zdecentralizowanego przechowywania danych, stworzenia zdecentralizowanej aplikacji definiującej uprawnienia do odczytu danych oraz aplikacji odpowiadającej za ich bezpieczne przetwarzanie. Ze względu na rozproszony charakter, usługa powinna wykorzystywać prosty interfejs, działający wyłączenie na urządzeniu użytkownika. System może zostać kompletnie zdefiniowany przez następujące wymagania funkcjonalne:

WF1. Tworzenie szyfrogramów plików.
WF2. Umieszczanie plików z wykorzystaniem zdecentralizowanej usługi przechowywania danych.
WF3. Ustalanie reguł dostępu do pliku.
WF4. Przeszyfrowywanie danych dla docelowego odbiorcy.
WF5. Pobieranie danych ze wskazanej lokalizacji i ich dekryptaż.

Powyższe wymagania funkcjonalne są uzupełniane przez następujące wymagania niefunkcjonalne:

WNF1. Użytkownik korzysta z dedykowanej aplikacji przeglądarkowej.
WNF2. Wykorzystanie uprzednio wdrożonego kontraktu sieci Ethereum do uwierzytelniania użytkowników.

WNF3. Składowanie danych z wykorzystaniem usługi zdecentralizowanej.

WNF4. System zapewnia ochronę kryptograficzną plików na poziomie 128 bitów.

WNF5. System zapewnia dostępność 24/7 z minimalnym czasem przestoju spowodowanym awariami zależnych sieci.

WNF6. System zapewnia dostęp do treści tylko dla upoważnionych użytkowników.

System nie przewiduje występowania ról użytkowników. Każdy z uczestników posiada możliwość realizacji zdefiniowanych przypadków użycia. W podstawowej implementacji systemu, zakładającej uprzednie utworzenie smart kontraktu odpowiedzialnego za wskazywanie upoważnionych do dekryptażu danej treści, można wyróżnić dwa przypadki użycia:

PU1. Umieszczenie nowej treści w systemie.

PU2. Pobranie i dekryptaż treści przechowywanej w systemie.

Szczegółowy opis przypadków użycia został przedstawiony w poniższych tabelach.

Tabela 5 Scenariusz przypadku użycia PU1

Nazwa przypadku użycia	PU1 Umieszczenie nowej treści w systemie
Warunki początkowe	Użytkownik chce zamieścić nową treść w systemie. System jest aktywny, użytkownik posiada dostęp do aplikacji końcowej.
Scenariusz	1. Użytkownik wybiera treść, która ma zostać udostępniona. 2. Aplikacja generuje jednorazowy, symetryczny klucz kryptograficzny dla treści. 3. Aplikacja szyfruje dane z wykorzystaniem usług WebCrypto API. 4. Aplikacja szyfruje jednorazowy klucz z wykorzystaniem klucza publicznego kontraktu i przesyła ten szyfrogram wraz z zaszyfrowanymi danymi do IPFS. 5. Użytkownik otrzymuje CID umieszczonego zasobu.
Wyniki końcowe	Nowa treść została umieszczona w sieci IPFS i przypisany został jej identyfikator CID.
Powiązania z innymi przypadkami użycia	Brak
Dodatkowe informacje	Prymitywy kryptograficzne zostały zdefiniowane w dalszej części pracy.

Tabela 6 Scenariusz przypadku użycia PU2

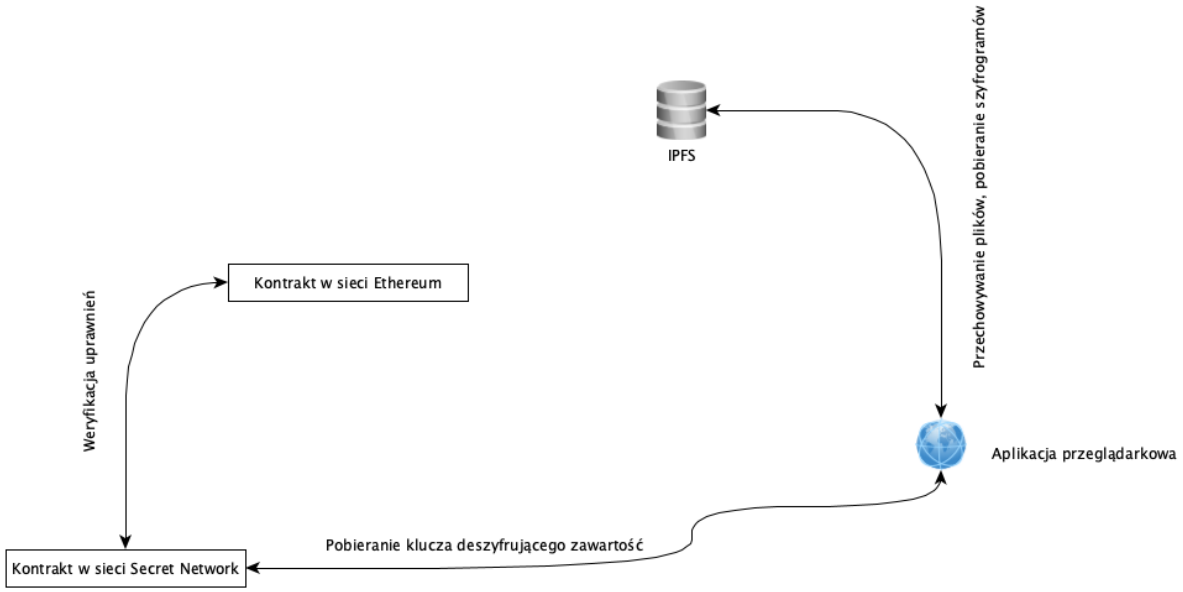
Nazwa przypadku użycia	PU2 Pobranie i dekryptaż treści przechowywanej w systemie
Warunki początkowe	Użytkownik chce uzyskać dostęp do treści przechowywanej w systemie. Reguły dostępu do danych zostały zdefiniowane.
Scenariusz	1. Użytkownik deklaruje adres CID zasobu, do którego chce uzyskać dostęp. 2. Użytkownik przesyła uzyskany szyfrogram klucza symetrycznego do kontraktu reeencryption proxy, potwierdzając swoją tożsamość. 3. Kontrakt weryfikuje uprawnienia użytkownika do odszyfrowania klucza i jeżeli weryfikacja jest pozytywna, przesyła odszyfrowany klucz dostępu do danych zaszyfrowany kluczem publicznym użytkownika. 4. Użytkownik pobiera szyfrogram treści. 5. Użytkownik dokonuje dekryptażu klucza deszyfrującego, a następnie odszyfrowuje pobraną zawartość.
Wyniki końcowe	Uprawniony użytkownik uzyskał dostęp do zaszyfrowanej treści.
Powiązania z innymi przypadkami użycia	Brak
Dodatkowe informacje	Prymitywy kryptograficzne zostały zdefiniowane w dalszej części pracy.

Architektura systemu zakłada wykorzystanie czterech elementów odpowiedzialnych za realizację założonych wymagań funkcjonalnych, niefunkcjonalnych oraz zdefiniowanych przypadków użycia. Elementy zostały wymienione poniżej:

- IPFS pełniący rolę zdecentralizowanej usługi przechowywania danych.
- Smart kontrakt sieci Ethereum odpowiadający za kontrolę dostępu do danych.
- Smart kontrakt sieci Secret Network realizujący funkcjonalność zdecentralizowanego reencryption proxy.
- Aplikacja przeglądarkowa odpowiedzialna za komunikację ze smart kontraktami i siecią IPFS.

IPFS jest wykorzystywane w sposób bierny, jedynie do składowania przesłanych danych. Ze względu na przyjęte założenia projektowe, pliki składające się na przechowywaną treść wraz z odpowiadającym im, zaszyfrowanym kluczem kryptograficznym, są przechowywane w ramach wspólnego katalogu. Powyższe ma na celu uproszczenie schematu adresowania zasobów w sieci IPFS. CID wskazuje katalog zawierający zbiór plików, a pliki mogą być lokalizowane z wykorzystaniem nazw. Tak przyjęty sposób przechowywania danych pozwala na zachowanie elastyczności w zakresie przechowywanych treści. Mogą to być zarówno

pojedyncze pliki, strumienie multimediów zgodne z formatem HLS [176], a także statyczne strony WWW.



Rysunek 31 Architektura systemu

Kontrakt sieci Ethereum może implementować dowolną logikę biznesową wymaganą od aplikacji. W szczególności, kontrakt może weryfikować posiadanie przez użytkownika określonego NFT (ERC-721), faktu dokonania wpłaty określonej sumy tokenów w zadanym okresie lub występowanie danego adresu w przestrzeni danych kontraktu. Poniżej przedstawiony został kod źródłowy kontraktu, który posiada jedną funkcję weryfikującą posiadanie przez wskazany adres danego NFT. Oznaczać to może przypadek użycia, w którym użytkownik zakupił dostęp do danej treści i udowadnia swoje prawo własności poprzez posiadanie odpowiedniego tokenu NFT. Kontrakt implementuje funkcję `verifyOwnership`, która zwraca wartość prawda lub fałsz w zależności od wyniku przeprowadzonej weryfikacji.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

interface IERC721 {
    function ownerOf(uint256 tokenId) external view returns (address owner);
}

contract NFTOwnershipChecker {
    function verifyOwnership(address nftContract, uint256 tokenId, address
owner) public view returns (bool) {
        IERC721 nft = IERC721(nftContract);
        return nft.ownerOf(tokenId) == owner;
    }
}
```

Poniższy kod kontraktu realizuje funkcjonalność przechowania listy uprawnionych adresów, gdzie każdy z adresów może być dodany jedynie przez właściciela kontraktu. Pozwala to na realizację przypadku użycia, w którym właściciel kontraktu – DAO, osoba fizyczna, osoba prawna, inny smart kontrakt decyduje o przyznaniu lub usunięciu uprawnień dostępu do danej treści. Kontrakt ten posiada funkcję `hasAccess` zwracającą wartość prawda lub fałsz w zależności od wyniku weryfikacji.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract AccessControl {
    mapping(address => bool) private accessList;
    address public owner;

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can perform this
action");
        _;
    }

    constructor() {
        owner = msg.sender;
    }

    function grantAccess(address _address) public onlyOwner {
        accessList[_address] = true;
    }

    function revokeAccess(address _address) public onlyOwner {
        accessList[_address] = false;
    }

    function hasAccess(address _address) public view returns (bool) {
        return accessList[_address];
    }
}
```

Smart kontrakt sieci Secret Network ma na celu realizację operacji przeszyfrowywania danych dla dedykowanego odbiorcy (reencryption proxy). Wybór stosowanych w tym celu prymitywów kryptograficznych podyktowany jest założeniami projektowymi przyjętymi w sieci Ethereum. Sieć ta, podobnie jak Bitcoin wykorzystuje krzywą secp256k1 [177]. Krzywa ta jest głównie wykorzystywana przez sieć do wykonywania podpisów transakcji w sieci. Uczestnicy sieci wykorzystują parę kluczy secp256k1 do uwierzytelniania wykonywanych przez siebie czynności. Adresy użytkowników sieci są generowane na podstawie skrótu Keccak-256 [178] klucza publicznego użytkownika. Proponowany w niniejszej rozprawie smart kontrakt realizuje operację przeszyfrowania danych z wykorzystaniem schematu ECIES (Elliptic Curve Integrated Encryption Scheme) [179], wykorzystującego krzywą secp256k1.

Zdefiniujmy następujące dwie funkcje:

- $Enc: K \times M \rightarrow C$, która przypisuje każdej parze klucza publicznego K i tekstu jawnego M odpowiadający im szyfrogram C,
- $Dec: K \times C \rightarrow M$, która przypisuje każdej parze klucza prywatnego K i szyfrogramu C odpowiadający im tekst jawny M.

Proces realizowany przez smart kontrakt sieci Secret Network może zostać zdefiniowany jako następujące odwzorowanie:

$$R: C \times SK \times PK \rightarrow C'$$

gdzie C' jest nowym szyfrogramem zaszyfrowanym dla innego odbiorcy, SK oznacza klucz prywatny, PK klucz publiczny.

Odwzorowanie R można zdefiniować jako:

$$R(C_k, SK_{contract}, PK_{rcv}) = Enc(PK_{rcv}, Dec(SK_{contract}, C_k))$$

gdzie:

- C_k oznacza szyfrogram zaszyfrowany kluczem publicznym kontraktu,
- PK_{rcv} oznacza klucz publiczny odbiorcy,
- $SK_{contract}$ oznacza klucz prywatny kontraktu.

Zapis $y = R(C_k, SK_{contract}, PK_{rcv})$ oznacza, że biorąc szyfrogram C_k, który zaszyfrowano przy użyciu klucza publicznego kontraktu, deszyfrujemy go przy użyciu klucza prywatnego $SK_{contract}$ kontraktu, a następnie szyfrujemy wynikającą wiadomość używając klucza publicznego PK_{rcv} odbiorcy, otrzymując C'_k jako wynik.

Kontrakty sieci Secret Network są tworzone z wykorzystaniem języka programowania Rust, a następnie kompilowane do postaci kodu CosmWasm. Skompilowany kod może zostać wdrożony w sieci. Ze względu na poziom skomplikowania kontraktu, poniżej zostały przedstawione najbardziej istotne fragmenty kodu źródłowego realizujące założoną funkcjonalność. **Poniższy fragment kodu przedstawia funkcje wykorzystywane w procesie inicjalizacji kontraktu oraz przeszyfrowywania zawartości**.

```
use core::mem;
use aes::cipher::generic_array::GenericArray;
use cosmwasm_std::{ Env, HumanAddr, Storage};
use libsecp256k1::{Error, PublicKey, SecretKey, util::FULL_PUBLIC_KEY_SIZE};
use secret_toolkit::crypto::{Prng, sha_256};
use ctr::cipher::{NewCipher, StreamCipher};
use hmac::{Hmac, Mac, NewMac};
```

```
…
pub fn generate_key_id(env: &Env, seed: &[u8]) -> String {
    let entropy = unsafe { mem::transmute::<u64, [u8; 8]>(env.block.height)
};

    let mut rng = Prng::new(seed, entropy.as_ref());
    rng.set_word_pos(0);
    format!("key_{}", base64::encode(rng.rand_bytes()))
}

pub fn generate_seed(keying_material: &str) -> [u8; 32] {
    sha_256(&keying_material.as_bytes())
}

pub fn generate_private_key(env: &Env, seed: &[u8], entropy: &[u8]) -> [u8;
32] {
    let    height_slice    =    unsafe    {    mem::transmute::<u64,    [u8;
8]>(env.block.height) };

    let          mut          keying_material:          Vec<u8>          =
env.message.sender.0.as_bytes().to_vec();
    keying_material.extend_from_slice(height_slice.as_ref());
    keying_material.extend_from_slice(entropy);

    let mut rng = Prng::new(seed, &keying_material);
    rng.set_word_pos(0);
    rng.rand_bytes()
}
pub fn encrypt(receiver_pub: &[u8], msg: &[u8]) -> Result<Vec<u8>, Error> {
    let receiver_pk = PublicKey::parse_slice(receiver_pub, None)?;
    let (ephemeral_sk, ephemeral_pk) = generate_keypair(receiver_pub, msg);

    let aes_key = encapsulate(&ephemeral_sk, &receiver_pk)?;
    let enc_key = &aes_key[..16];
    let            encrypted            =            aes_encrypt(&enc_key,
msg).ok_or(Error::InvalidMessage)?;
    let mac_key = sha_256(&aes_key[16..]);
    let default_mac: [u8; 2] = [0, 0];
    let  mut  data_mac:  Vec<u8>  =  Vec::with_capacity(encrypted.len()  +
default_mac.len());
    data_mac.extend(&encrypted);
    data_mac.extend(default_mac);

    let mut mac = HmacSha256::new_from_slice(&mac_key)
        .expect("HMAC can take key of any size");
    mac.update(&data_mac);

    let result = mac.finalize();

    let mac_bytes = result.into_bytes();

    let  mut  cipher_text  =  Vec::with_capacity(FULL_PUBLIC_KEY_SIZE  +
encrypted.len() + mac_bytes.len());
    cipher_text.extend(ephemeral_pk.serialize().iter());
    cipher_text.extend(encrypted);
    cipher_text.extend(mac_bytes);

    Ok(cipher_text)
}
pub fn decrypt(receiver_sec: &[u8], msg: &[u8]) -> Result<Vec<u8>, Error> {
    let receiver_sk = SecretKey::parse_slice(receiver_sec)?;
```

```
    if msg.len() < FULL_PUBLIC_KEY_SIZE {
        return Err(Error::InvalidMessage);
    }

    let ephemeral_pk = PublicKey::parse_slice(&msg[..FULL_PUBLIC_KEY_SIZE],
None)?;
    let encrypted = &msg[FULL_PUBLIC_KEY_SIZE..msg.len()-32];
    let mac_bytes = &msg[msg.len()-32..];
    let aes_key = decapsulate(&ephemeral_pk, &receiver_sk)?;
    let mac_key = sha_256(&aes_key[16..]);
    let enc_key = &aes_key[..16];
    let default_mac: [u8; 2] = [0, 0];
    let  mut  data_mac:  Vec<u8>  =  Vec::with_capacity(encrypted.len()  +
default_mac.len());
    data_mac.extend(encrypted);
    data_mac.extend(default_mac);
    let mut mac = HmacSha256::new_from_slice(&mac_key)
        .expect("HMAC can take key of any size");
    mac.update(&data_mac);
    mac.verify(&mac_bytes).unwrap();
    aes_decrypt(&enc_key, encrypted).ok_or(Error::InvalidMessage)
}

pub  fn  generate_keypair(seed:  &[u8],  entropy:  &[u8])  ->  (SecretKey,
PublicKey) {
    let mut rng = Prng::new(seed, entropy);
    rng.set_word_pos(0);
    let sk = SecretKey::parse(&rng.rand_bytes()).unwrap();
    (sk.clone(), PublicKey::from_secret_key(&sk))
}
pub fn decapsulate(pk: &PublicKey, peer_sk: &SecretKey) -> Result<AesKey,
Error> {
    let mut shared_point = pk.clone();
    shared_point.tweak_mul_assign(&peer_sk)?;
    let mut master = Vec::with_capacity(32);
    master.extend(shared_point.serialize_compressed()[1..].iter());
    let hk = kdf_sha256(master.as_slice())?;

    Ok(hk)
}
```

Powyższy kod realizuje podstawowe funkcjonalności kryptograficzne kontraktu, w szczególności generację kluczy kryptograficznych i realizację ECIES. Funkcjonalności te są udostępniane poprzez następujący kod:

```
pub fn query<S: Storage, A: Api, Q: Querier>(
    deps: &Extern<S, A, Q>,
    msg: QueryMsg,
) -> StdResult<Binary> {
    let result: QueryResponse = match msg {
        QueryMsg::PublicKey {
            key_id
        }=> {
            let record = load_key_record(&deps.storage, &key_id)?;

            let private_key = SecretKey::parse(&record.key).unwrap();
            let public_key = PublicKey::from_secret_key(&private_key);

            let public_key = public_key.serialize_compressed();
```

```
            PublicKeyResponse { public_key }.into()
        },
        QueryMsg::Reencrypt {
            address,
            key_id,
            data,
        }=> {
            // ------------------ Decryption (stage I) ---------------
            // Key
            let record = load_key_record(&deps.storage, &key_id)?;

            // Decrypt data using the secret key of contract
            let data_bytes = hex::decode(data).map_err(|_| {
                StdError::generic_err("Error validating data format: should
be hex string")
            })?;

            let data_arr = data_bytes.as_slice();
            let temp = decrypt(&record.key, data_arr).unwrap();
            let data = temp.as_slice();

            //------------------- Encryption (stage II) -------------
            // Key
            let        state        =        get_subscription(&deps.storage,
address.as_bytes())?;

            if !state.state {
                return  Err(StdError::generic_err("Error  address  is  not
subscribed"));
            }

            let userpubkey_bytes = hex::decode(state.public_key).map_err(|_|
{
                StdError::generic_err("Error validating public key format:
should be hex string")
            })?;

            let mut userpk_arr = [0u8; 33];
            userpk_arr.copy_from_slice(&userpubkey_bytes);
            let                          user_pubkey                          =
PublicKey::parse_compressed(&userpk_arr).unwrap();
            let public_key = user_pubkey.serialize_compressed();

            let temp = encrypt(&public_key, &data).unwrap();
            let ciphertxt = temp.as_slice();

            ReencryptResponse { ciphertxt }.into()
        },
        QueryMsg::Encrypt {
            key_id,
            data,
        }=> {

            // ------------------ Encryption (stage I) ---------------
            // Key
            let record = load_key_record(&deps.storage, &key_id)?;

            let data_bytes = data.as_bytes();
```

```
            let                          public_key                          =
PublicKey::from_secret_key(&SecretKey::parse(&record.key).unwrap());
            let public_key = public_key.serialize_compressed();

            // Encrypt
            let temp = encrypt(&public_key, &data_bytes).unwrap();
            let ciphertxt = temp.as_slice();

            EncryptResponse { ciphertxt }.into()
        },
        QueryMsg::Decrypt {
            key_id,
            data,
        }=> {

            // ------------------ Decryption (stage II) ---------------
            // Key
            let record = load_key_record(&deps.storage, &key_id)?;

            let data_bytes = hex::decode(data).map_err(|_| {
                    StdError::generic_err("Error  validating  data  format:
should be hex string")
                })?;

            let data_arr = data_bytes.as_slice();

            // Decrypt
            let temp = decrypt(&record.key, &data_arr).unwrap();
            let plaintxt = temp.as_slice();

            DecryptResponse                    {                    result:
std::str::from_utf8(&plaintxt).unwrap().to_string() }.into()
        }
        QueryMsg::IsSubscribed {
            address
        } => {
            let        state        =        get_subscription(&deps.storage,
address.as_bytes())?;
            QueryResponse {
                messages: format!("{}", state.state),
            }
        },
        QueryMsg::GetRequest {} => {
            let res_queue = get_req_queue(&deps.storage)?.queue;

            let result = if res_queue.len() == 0 {
                QueryResponse {
                    messages: "0".to_string()
                }
            } else {
                QueryResponse {
                    messages:    format!("{}    {:#?}",    res_queue.len(),
res_queue[0]),
                }
            };
            result
        }
    };
    to_binary(&Ok(result)?)
}
```

Aplikacja przeglądarkowa pełni rolę interfejsu komunikacyjnego pomiędzy siecią IPFS, kontraktem sieci Secret Network przekazującym klucze kryptograficzne a użytkownikiem. Realizacja założonych przypadków użycia wymaga implementacji w aplikacji kodu odpowiedzialnego za szyfrowanie plików z wykorzystaniem algorytmu symetrycznego, szyfrowanie klucza z wykorzystaniem schematu ECIES, przesłanie plików do sieci IPFS oraz wykonanie operacji deszyfrowania danych pozyskanych z sieci. Realizacja szyfrowania symetrycznego może zostać zaimplementowana z wykorzystaniem WebCryptography API [180]. Powyższe pozwala na realizację operacji szyfrowania w czasie zbliżonym do implementacji natywnych, gwarantując wysoką wydajność rozwiązania. Poniższy kod źródłowy implementuje zakładaną funkcjonalność szyfrowania i deszyfrowania plików, z wykorzystaniem AES-GCM-256:

```
    async function encryptfile() {
        btnEncrypt.disabled=true;
        btnDecrypt.disabled=true;

        var plaintextbytes=await readfile(objFile)
        .catch(function(err){
            console.error(err);
        });
        var plaintextbytes=new Uint8Array(plaintextbytes);

        //key exportable=true
        window.crypto.subtle.generateKey( { name: "AES-GCM", length: 256 },
true,  ["encrypt", "decrypt"] )
.then(function(key){
                    window.crypto.subtle.exportKey(
                "jwk",
                key
                )
                .then(function(keydata){
                localStorage.setItem("tempkey",JSON.stringify(keydata));
                })
                .catch(function(err){
                console.error(err);
                });
                var iv  = window.crypto.getRandomValues(new Uint8Array(12))
                window.crypto.subtle.encrypt({  name:  "AES-GCM",iv:  iv,},
key, plaintextbytes )
            .then(function(cipherbytes){
                    if(!cipherbytes) {
                        spnEncstatus.classList.add("redspan");
                        spnEncstatus.innerHTML='<p>Error  encrypting  file.
See console log.</p>';
                        return;
                    }

                    var            resultbytes            =            new
Uint8Array(cipherbytes.byteLength+12)
                    cipherdata = new Uint8Array(cipherbytes);
                    resultbytes.set(iv, 0);
                    resultbytes.set(cipherdata, 12);
```

```
                    var      blob=new      Blob([resultbytes],      {type:
'application/download'});
                    var blobUrl=URL.createObjectURL(blob);
                    aEncsavefile.href=blobUrl;
                    aEncsavefile.download=objFile.name + '.enc';

                    spnEncstatus.classList.add("greenspan");
                    spnEncstatus.innerHTML='<p>File encrypted.</p>';
                    aEncsavefile.hidden=false;
                }).catch(function(err){
                console.error(err);
            });
        }).catch(function(err){
            console.error(err);
        });
    }

    async function decryptfile() {
        btnDecrypt.disabled=true;
        btnEncrypt.disabled=true;

        var cipherbytes=await readfile(objFile)
        .catch(function(err){
            console.error(err);
        });
        var cipherbytes=new Uint8Array(cipherbytes);
        var ivbytes=cipherbytes.slice(0,12);
        cipherbytes=cipherbytes.slice(12);
        var keydata=localStorage.getItem('tempkey');
        if(keydata!=null)
        keydata = JSON.parse(localStorage.getItem('tempkey'));
        console.log('IV and key imported');
        window.crypto.subtle.importKey(
                "jwk",
                {
        kty: keydata.kty,
        k:   keydata.k,
        alg: keydata.alg,
        ext: keydata.ext
            },
                {
                        name: "AES-GCM",
                },
                false,
                ["encrypt", "decrypt"]
        )
        .then(function(key){
                window.crypto.subtle.decrypt(
            {
                name: "AES-GCM",
                iv: ivbytes
            },
            key,
            cipherbytes
                )
                .then(function(plaintextbytes){
                            if(!plaintextbytes) {
                                spnDecstatus.classList.add("redspan");
                                spnDecstatus.innerHTML='<p>Error  decrypting
file.  Key may be incorrect.</p>';
                                return;
```

```
                            }
                            plaintextbytes=new Uint8Array(plaintextbytes);

                            var   blob=new   Blob([plaintextbytes],   {type:
'application/download'});
                            var blobUrl=URL.createObjectURL(blob);
                            aDecsavefile.href=blobUrl;
                            aDecsavefile.download=objFile.name + '.dec';

                            spnDecstatus.classList.add("greenspan");
                            spnDecstatus.innerHTML='<p>File
decrypted.</p>';
                            aDecsavefile.hidden=false;
                    })
                    .catch(function(err){
                        console.error(err);
                });
            })
            .catch(function(err){
    console.error(err);
});
    }
```

Proces szyfrowania klucza i wartości wektora inicjalizującego (IV), w celu ich przechowania w sieci IPFS jest wykonywany z wykorzystaniem algorytmu ECIES, szyfrującego dane z wykorzystaniem klucza publicznego kontraktu sieci Secret Network.

Zdefiniujmy funkcję szyfrowania Enc jako:

$Enc: K \times M \rightarrow C$, która przypisuje każdej parze klucza publicznego K i tekstu jawnego M odpowiadający im szyfrogram C,

Odwzorowanie Enc można zdefiniować jako:

$$Enc(PK_{contract}, (K, IV))$$

gdzie:

- $PK_{contract}$ oznacza klucz publiczny kontraktu,
- K oznacza klucz symetryczny AES-GCM-256,
- IV oznacza wektor inicjalizujący dla AES-GCM-256.

Zapis $C_k = E(PK_{contract}, (K, IV))$ oznacza, że klucz symetryczny K wraz z wektorem inicjalizującym IV jest szyfrowany przy użyciu klucza publicznego $PK_{contract}$ dając jako wynik szyfrogram C_k, który jest bezpiecznie przechowywany.

Proces deszyfrowania klucza symetrycznego i wektora IV pozyskanego z sieci IPFS jest przeprowadzany w sposób analogiczny. Zdefiniujmy funkcję deszyfrowania Dec jako:

$Dec: K \times C \rightarrow M$, która przypisuje każdej parze klucza prywatnego K i szyfrogramu C odpowiadający im tekst jawny M, w tym przypadku par klucz symetryczny wektor IV.

Odwzorowanie Dec można zdefiniować jako:

$$(K, IV) = Dec(SK_{rcv}, C'_k)$$

gdzie:

- C'_k oznacza klucz symetryczny AES wraz z wektorem IV zaszyfrowany kluczem publicznym użytkownika, przeszyfrowany przez kontrakt sieci Secret Network,
- SK_{rcv} oznacza klucz prywatny użytkownika,

Zapis ten oznacza, że szyfrogram C'_k jest deszyfrowany przy użyciu klucza prywatnego SK_{rcv} dając jako wynik parę oryginalnego klucza symetrycznego K i wektora inicjalizującego IV.

Powyższe może zostać zrealizowane z wykorzystaniem kodu JavaScript rozszerzonego o wykorzystanie dodatkowych bibliotek – elliptic [181] oraz eth-crypto [182]. Następujący kod pozwala na realizację operacji ECIES z wykorzystaniem wymienionych bibliotek:

```
import { ec as EC } from 'elliptic';
import EthCrypto from 'eth-crypto';

const ec = new EC('secp256k1');
const keyPair = ec.genKeyPair();
const publicKey = keyPair.getPublic('hex');
const privateKey = keyPair.getPrivate('hex');

async function encryptWithPublicKey(publicKey, message) {
    const encrypted = await EthCrypto.encryptWithPublicKey(publicKey,
message);
    return EthCrypto.cipher.stringify(encrypted);
}

async function decryptWithPrivateKey(privateKey, encrypted) {
    const cipher = EthCrypto.cipher.parse(encrypted);
    const decrypted = await EthCrypto.decryptWithPrivateKey(privateKey,
cipher);
    return decrypted;
}
```

Komunikacja z siecią IPFS może zostać wykonana z użyciem biblioteki ipfs-http-client [183] do umieszczania danych w sieci oraz z wykorzystaniem bramki http, która pozwala na pobieranie materiałów z sieci IPFS w sposób nie różniący się od pobierania danych z

dowolnego serwera sieci web. Poniższy kod źródłowy implementuje funkcjonalność przesyłania danych do sieci IPFS i umieszczania ich we wspólnym katalogu:

```
const uploadWithFileName = async (newFile) => {
  if (newFile && newFile.name) {
    const fileDetails = {
      path: newFile.name,
      content: newFile,
    };
    const options = {
      wrapWithDirectory: true,
    };

    try {
      const added = await ipfs?.add(fileDetails, options);
      return added;
    } catch (err) {
      console.error(err);
    }
  }
};
```

Komunikacja z kontraktem sieci Secret Network może zostać zrealizowana poprzez wykorzystanie biblioteki secretjs [184]. Aby komunikacja była możliwa, użytkownik musi posiadać możliwość podpisywania transakcji w sieci. Biblioteka secretjs pozwala na utworzenie klienta wewnątrz aplikacji przeglądarkowej, umożliwiając pełną integrację z siecią. Poniżej przedstawiony został fragment kodu źródłowego, który pozwala na utworzenie klienta sieci oraz przesłanie żądania przeszyfrowania klucza do wskazanego zasobu.

```
const {
  EnigmaUtils,
  Secp256k1Pen,
  SigningCosmWasmClient,
  pubkeyToAddress,
  encodeSecp256k1Pubkey,
} = require("secretjs");
…
  const signingPen = await Secp256k1Pen.fromMnemonic(mnemonic);
  const pubkey = encodeSecp256k1Pubkey(signingPen.pubkey);
  const accAddress = pubkeyToAddress(pubkey, "secret");
  const txEncryptionSeed = EnigmaUtils.GenerateNewSeed();
  const client = new SigningCosmWasmClient(
    httpUrl,
    accAddress,
    (signBytes) => signingPen.sign(signBytes),
    txEncryptionSeed,
    customFees
  );
…
  response = await client.queryContractSmart(contractAddress, {
    Encrypt: {
      address: ethUserAddress,
      key_id: "KEY_ID",
      user_public_key: public_key,
      data: ciphertext,
    },
```

```
});
```

Realizacja przypadków użycia PU1 oraz PU2 jest osiągana poprzez implementację powyższych schematów, w odpowiadających aplikacjach i sieciach. Zaproponowane kody źródłowe mogą stanowić podstawę do implementacji innych przypadków użycia, w których możliwe jest dodanie lub usunięcie wybranych elementów systemu. Realizacja uproszczonego schematu dostępu jest możliwa poprzez wykorzystanie jedynie smart kontraktu w sieci Secret Network, co pozwoli na zmniejszenie ograniczeń implementacji związanych z komunikacją pomiędzy blockchainami, a także wynikających z powyższego wyzwań implementacyjnych. W szczególności należy zauważyć, iż w obecnej formule wymagane jest wykorzystanie trzech różnych języków programowania – Rust, Solidity oraz JavaScript, w celu wytworzenia kompletnego rozwiązania. Wyeliminowanie kontraktu Ethereum pozwala na eliminację wykorzystania języka Solidity i uproszczenie architektury. Ostateczny wybór elementów składowych systemu powinien zostać dopasowany do konkretnego scenariusza biznesowego, jednakże ogólna konstrukcja schematu uwierzytelnionego dostępu może nadal być wykorzystywana.

Podsumowując, zaprezentowany schemat **stanowi oryginalne i innowacyjne rozwiązanie w zakresie autoryzacji danych w środowiskach rozproszonych, wykorzystując smart kontrakty w sieciach blockchain**. **Autorskie implementacje w sieci Secret Network i Ethereum pokazują, jak zaawansowane technologie kryptograficzne mogą być efektywnie wykorzystane do ochrony prywatności i bezpieczeństwa danych**. Proponowany system, opierając się na zdecentralizowanym przechowywaniu danych i reencryption proxy, **umożliwia bezpieczne zarządzanie dostępem do zasobów cyfrowych, co stanowi istotny wkład w rozwój technologii blockchain w kontekście bezpieczeństwa informacji**. Osiągnięte wyniki podkreślają znaczenie innowacyjności i adaptacji nowoczesnych technologii w realizacji skutecznych i bezpiecznych systemów informatycznych.

6.3. Ewaluacja proponowanego schematu

W podrozdziale tym przeprowadzono **dokładną ocenę zaproponowanego, autorskiego schematu uwierzytelnionego dostępu do danych w środowisku zdecentralizowanym**. Analiza skupia się na **ocenie integralności, poufności oraz dostępności przechowywanych danych**, zwracając uwagę na **skuteczność zastosowanych mechanizmów kryptograficznych oraz teoretyczne podatności systemu**.

Definicja 6.1.

Poufność to zdolność systemu do zapewnienia, że informacje są dostępne wyłącznie dla uprawnionych użytkowników.

Definicja 6.2.

Integralność to zdolność systemu do wykrywania i zapobiegania nieautoryzowanym zmianom danych.

Definicja 6.3.

Siła kryptograficzna, określana również jako poziom bezpieczeństwa, to miara odporności funkcji kryptograficznej na naruszenia, wyrażana bitach, gdzie bezpieczeństwo n-bitowe oznacza, że atakujący musiałby wykonać 2^n operacji, aby złamać dany algorytm.

Zaproponowany schemat realizacji zdecentralizowanego, uwierzytelnionego dostępu do danych z wykorzystaniem kontraktów sieci Secret Network, Ethereum oraz przechowywania danych w IPFS zakłada zachowanie poufności i integralności przechowywanych danych, a także udzielanie dostępu jedynie uprawnionym do tego podmiotom. Zgodnie z przyjętymi założeniami niefunkcjonalnymi, system powinien cechować się poziomem siły kryptograficznej co najmniej 128 bitów. W niniejszym rozdziale przedstawione zostały wyniki ewaluacji dla podstawowych usług kryptograficznych – poufności i integralności, a także dostępności danych. Przeprowadzona została także analiza teoretycznych podatności schematu, wraz z przedstawieniem możliwych technik ich mitygacji.

Zaproponowany w niniejszej pracy schemat pozwala na zapewnienie integralności następujących danych:

- Przechowywany w IPFS szyfrogram danych.
- Przechowywany w IPFS szyfrogram pary *(K,IV)* – klucz oraz wektora inicjalizujący dla algorytmu AES-256 GCM wykorzystywanego do szyfrowania danych.
- Pary *(K,IV)* przesłanej w szyfrogramie C'_k.

- Postać jawna udostępnianych przez schemat danych.
- Dane smart kontraktów.

Integralność danych przechowywanych w IPFS – szyfrogramów klucza symetrycznego, wektora inicjalizującego oraz właściwej treści, jest gwarantowana poprzez mechanizmy protokołu IPFS. W procesie umieszczania danych w sieci IPFS, obliczany jest skrót kryptograficzny zawartości umieszczanej we wspólnym folderze. Skrót ten jest obliczany z wykorzystaniem funkcji skrótu SHA-256. Zgodnie z publikacją [185] siła kryptograficzna tej funkcji skrótu wynosi 128 bitów, co spełnia założone wymagania niefunkcjonalne. Para *(K,IV)* przeszyfrowana przez kontrakt sieci Secret Network jest zabezpieczona przed naruszeniem integralności poprzez wykorzystanie kodu HMAC-SHA256. Podstawą jego budowy jest zastosowanie funkcji skrótu SHA-256, która zgodnie z [185] posiada siłę kryptograficzną wynoszącą 128 bitów. Szyfrogram danych, przechowywany w IPFS, powstaje w wyniku zaszyfrowania przechowywanej zawartości z wykorzystaniem algorytmu AES-256 pracującego w trybie GCM (Galois Counter Mode) [186]. Tryb ten gwarantuje zachowanie integralności danych z siłą kryptograficzną przewyższającą 128 bitów. Ostatni ze zidentyfikowanych typów danych, to dane przechowywane przez kontrakty sieci Ethereum i Secret Network. Integralność danych przechowywanych w kontraktach wynika z połączenia dwóch czynników – skali sieci oraz wykorzystywanych kryptograficznych funkcji skrótu. W przypadku sieci Ethereum wykorzystywana jest funkcja Keccak-256, posiadająca siłę kryptograficzną na poziomie 128 bitów, dla sieci Secret Network jest to SHA-256 posiadające taki sam poziom siły kryptograficznej. Obydwie z wymienionych sieci są wrażliwe na atak 51% mocy [187]. Ze względu na wykorzystanie algorytmu konsensusu typu Proof-of-Stake, atak taki wymaga zgromadzenia ponad 51% zasobów sieci. Bazując na [187], atak taki nie jest praktycznie wykonywalny, a jego charakter nie pozwala na przypisanie wartości siły kryptograficznej do tego parametru. W związku z powyższym założono, że obydwie z wymienionych sieci gwarantują zakładany przez wymagania niefunkcjonalne parametr siły kryptograficznej. Założenie to zostało przyjęte na podstawie wykorzystywanych przez te sieci prymitywów kryptograficznych. Konkludując, schemat zapewnia integralność danych na zakładanym poziomie siły kryptograficznej wynoszącym 128 bitów.

Schemat zakłada przechowywanie danych z wykorzystaniem sieci publicznych, do których dostęp jest otwarty. Realizacja scenariusza uwierzytelnionego dostępu do treści wymaga zastosowania mechanizmów kryptograficznych, które pozwolą na zachowanie poufności

danych przechowywanych w IPFS. W tym kontekście zdefiniowane zostały następujące artefakty, dla których zastosowano mechanizmy zapewnienia poufności:

- Postać jawna udostępnianej przez schemat treści.
- Para *(K,IV)* wykorzystana do wytworzenie szyfrogramu przechowywanego w IPFS.
- Klucz prywatny kontraktu sieci Secret Network, wykorzystywany w procesie przeszyfrowywania.

Wymagania niefunkcjonalne zakładają zagwarantowanie poziomu siły kryptograficznej na poziomie 128 bitów. Postać jawna udostępnianej treści jest szyfrowana z wykorzystaniem algorytmu AES-256, pracującego w trybie GCM. Algorytm ten posiada siłę kryptograficzną większą niż zakładane 128 bitów, należy zatem przyjąć, iż w kontekście ochrony poufności treści jest to rozwiązanie wystarczające. Para *(K,IV)* jest przechowywana w postaci zaszyfrowanej. Szyfrogram jest generowany poprzez schemat ECIES, wykorzystujący krzywą secp256k1. Poziom siły kryptograficznej dla ECIES jest zdefiniowany przez zastosowaną krzywą i wynosi 128 bitów.

Bezpieczeństwo klucza prywatnego kontraktu sieci Secret Network jest zależne wyłącznie od mechanizmów oferowanych przez sieć. Zgodnie z [166] dane kontraktu są szyfrowane z wykorzystaniem AES-128-SIV, a klucz symetryczny wykorzystywany przez ten algorytm jest generowany z wykorzystaniem schematu ECDH (Elliptic Curve Diffie-Hellman) [188] wykorzystującego krzywą curve25519 [189]. Sekrety te są przechowywane w ramach Intel SGX, co oznacza ich ekspozycję na błędy w implementacji tej platformy. W roku 2022 opublikowany został atak na mechanizm SGX i sieć Secret Network [190] pozwalający na odzyskanie poufnych danych kontraktów. Twórcy sieci zaproponowali i wdrożyli zmiany implementacyjne, które wykluczają tą podatność [191]. Powyższy atak nie był atakiem na algorytm kryptograficzny, lecz na implementację bezpiecznej enklawy. Występowanie błędów programistycznych nie pozwala na oszacowanie poziomu siły kryptograficznej rozwiązania, lecz musi być brane pod uwagę przy wdrażaniu rozwiązania w konkretnym przypadku użycia. Zważywszy na powyższe, poziom siły kryptograficznej określony wykorzystanymi w systemie prymitywami kryptograficznymi wynosi 128 bitów i spełnia założone wymagania niefunkcjonalne.

Dostęp do danych przechowywanych w zaproponowanym schemacie jest możliwy wtedy i tylko wtedy, gdy wszystkie trzy wykorzystywane usługi – sieci IPFS, Ethereum oraz Secret

Network są dostępne. Ze względu na zdecentralizowany charakter tych sieci, ich całkowita niedostępność nie wystąpiła w całej historii działania. W przypadku sieci Ethereum i Secret Network użytkownicy mogą doświadczać problemów z dostępnością serwerów RPC. Ze względu na występowanie licznych dostawców tych usług, a także możliwości samodzielnego wdrożenia serwera RPC na własnej infrastrukturze, należy przyjąć, że zagrożenie wynikające z powyższego jest marginalne. Należy także zaznaczyć, iż ze względu na wprowadzony koszt komunikacji z kontraktem sieci Secret Network, przeprowadzenie ataku typu DoS lub DDoS staje się nieopłacalne.

Pliki przechowywane w sieci IPFS są dostępne tak długo, jak długo istnieje aktywny węzeł sieci posiadający je w swoich zasobach. Jeżeli dany plik jest rzadko żądany przez inne węzły sieci IPFS, to może on nie być wystarczająco zreplikowany w sieci. W takim przypadku niedostępność węzłów hostujących dany plik powoduje brak jego dostępności. Ryzyko to może być ograniczane poprzez wykorzystanie usług przechowywania danych w IPFS, oferujących tzw. pinning zasobów lub poprzez utrzymywanie własnych węzłów sieci. Każdy żądany materiał zostaje zapisany w lokalnej pamięci podręcznej węzła, tym samym zwiększając jego replikację. W przypadku osiągnięcia limitów wykorzystania przestrzeni dyskowej, materiały rzadko żądane mogą zostać usunięte z pamięci podręcznej, tym samym zmniejszając ich dostępność w sieci. W kontekście zaproponowanego schematu należy przyjąć działania zmierzające do zapewnienia dostępności danych, takie jak wykorzystanie zewnętrznych dostawców usług IPFS i replikacja danych na wielu węzłach. Proces pobierania danych może zostać usprawniony poprzez wykorzystanie bramek http pozwalających na zwiększenie prędkości przesyłania plików. Zastosowanie powyższych środków zaradczych pozwoli na zachowanie dostępności deklarowanej w wymaganiach niefunkcjonalnych narzuconych na zaproponowany schemat.

Przedstawione rozwiązanie zdecentralizowanego, uwierzytelnionego dostępu do danych zapewnia zachowanie poufności, integralności, a przy zastosowaniu właściwego wdrożenia także dostępności plików. Należy jednak wskazać na możliwość uzyskania nieautoryzowanego dostępu, w wyniku działań takich jak:

- Udostępnienie treści pozyskanych przez uprawnionego uczestnika stronom trzecim.
- Utratę dostępu do klucza prywatnego przez uprawnionego uczestnika.
- Kompromitacja sieci Secret Network skutkująca wyciekiem klucza prywatnego kontraktu.

Konstrukcja aplikacji przeglądarkowej pozwala użytkownikom na dostęp do danych odszyfrowanych i pozbawionych dodatkowych mechanizmów zabezpieczeń. Może to skutkować nieuprawnionym udostępnieniem tak pozyskanych plików stronom trzecim, które nie uzyskały stosownych uprawnień zapisanych w kontrakcie. Taki sposób udostępniania treści ma miejsce poza środowiskiem blockchain wykonującym kontrakty i nie może być wyeliminowany z ich wykorzystaniem. Ochrona treści po ich odszyfrowaniu może być zapewniona poprzez zastosowanie dodatkowych mechanizmów izolacji, np. otwierania dokumentów w dedykowanym, izolowanym środowisku. Niestety, nawet zastosowanie izolowanego środowiska nie pozwala na całkowitą eliminację ryzyka nieautoryzowanego udostępnienia danych. Nadal możliwe jest zastosowanie tzw. ataków niskiej technologii, tj. wykonania zdjęć materiałów, wykonanie nagrania audio/wideo z wykorzystaniem zewnętrznego urządzenia itp.

Utrata dostępu do klucza prywatnego oznacza brak wyłącznej kontroli użytkownika nad adresem zarejestrowanym w usłudze. Kontrakt nie posiada możliwości określenia prawowitego właściciela, a wszystkie decyzje przezeń podejmowane są oparte o prawidłowość dostarczanych przez użytkownika podpisów cyfrowych. W przypadku przejęcia kontroli nad kluczem prywatnym, atakujący będzie posiadał takie same możliwości dostępu do danych jak użytkownik, do którego dany klucz należał. Kontrakt nie posiada możliwości technicznej zabezpieczenia przed tego rodzaju atakami. Pełna odpowiedzialność za zachowanie poufności klucza prywatnego spoczywa na użytkowniku. Jest to założenie tożsame do przyjętego powszechnie w sieciach blockchain i stanowi ogólnie akceptowany standard.

Zagrożeniem niosącym największe ryzyko dla zaprojektowanego schematu jest kompromitacja sieci Secret Network. Bezpieczeństwo całego systemu jest oparte o możliwość przechowywania sekretów w blockchain i wykorzystywania ich do obliczeń prowadzonych przez smart kontrakt. Naruszenie poufności danych przechowywanych w sieci Secret Network, czy to przez wykorzystanie luk w zabezpieczeniach wykorzystywanej bezpiecznej enklawy (w momencie pisania tej rozprawy jest to Intel SGX), czy też atak na implementację protokołu, oznaczałoby całkowitą utratę możliwości kontroli procesu przeszyfrowywania danych. Dla analizowanego schematu oznacza to utratę możliwości realizacji założonych funkcjonalności w sposób zgodny z założonymi wymaganiami funkcjonalnymi i niefunkcjonalnymi. Twórcy sieci prowadzą intensywne prace zmierzające do ograniczenia tego ryzyka [191], między innymi poprzez zastosowanie technik szyfrowania homomorficznego.

Biorąc pod uwagę aktualny stan wiedzy, **zaproponowany schemat realizuje zakładane wymagania co do zapewnienia poufności, integralności i dostępności danych**. Identyfikowane zagrożenia, choć istotne, posiadają niewielkie prawdopodobieństwo wystąpienia, a ich ryzyko może zostać zminimalizowane lub wyeliminowane poprzez wprowadzenie nowych rozwiązań technologicznych oraz ewolucję projektowanego systemu.

7. Podsumowanie i wnioski.

Gwałtowny rozwój technologii obserwowany w ostatnich dekadach całkowicie zmienił sposób w jaki informacje są przechowywane, przetwarzane i wymieniane. Jedną z bardziej istotnych innowacji ostatnich lat jest technologia blockchain, która wprowadzając kryptowaluty zrewolucjonizowała sektor finansowy. W kolejnych iteracjach technologia ta otworzyła nowe możliwości dla zwiększenia bezpieczeństwa i prywatności danych w wielu obszarach. **Niniejsza rozprawa miała na celu zbadanie potencjału wykorzystania zaawansowanych technologii blockchain do wzmacniania odporności systemów informatycznych.** Szczególny nacisk został położony na opracowanie podstaw do utworzenia systemu zapewniającego uwierzytelniony dostęp do danych zarządzanych przez infrastrukturę zdecentralizowaną.

Celem pracy było nie tylko przedstawienie i zrozumienie teoretycznych podstaw dla działania różnych technologii blockchain, ale także identyfikacja wyzwań związanych z zapewnieniem bezpieczeństwa danych w różnych obszarach, w szczególności przy praktycznym zastosowaniu tej technologii do rozwoju nowych metod ochrony informacji. Poprzez wykonanie analizy różnych aspektów związanych z decentralizacją, uwierzytelnianiem i zapewnieniem integralności danych, praca dostarcza cennych wskazówek dotyczących wykorzystania możliwości technologii blockchain do tworzenia systemów odpornych na ataki zewnętrzne i wewnętrzne, a także zwiększenia prywatności użytkowników.

W przyjętym procesie badawczym, kluczowym elementem była postawienie hipotez oraz ich weryfikacja. Poniższe tabele prezentują założone hipotezy badawcze, a także wyniki weryfikacji wraz z ich uzasadnieniem.

Tabela 7 Wynik weryfikacji hipotezy szczegółowej nr 1

Hipoteza badawcza HS_1	Zastosowanie połączenia różnych technologii blockchain pozwala na stworzenie rozproszonej usługi szyfrowania danych.
Hipoteza alternatywna HS'_1	Zastosowanie połączenia różnych technologii blockchain **NIE** pozwala na stworzenie rozproszonej usługi szyfrowania danych.
Wynik weryfikacji	Hipotezę HS_1 należy przyjąć.
Uzasadnienie	W pracy przedstawiony został schemat łączący sieć Ethereum oraz Secret Network pozwalający na realizację operacji szyfrowania danych dla użytkowników autoryzowanych przez smart kontrakt. Wykorzystanie

	kontraktu wdrożonego w sieci Secret Network pozwoliło na osiągnięcie założonego rozproszenia usługi.

Tabela 8 Wynik weryfikacji hipotezy szczegółowej nr 2

Hipoteza badawcza HS_2	Technologia blockchain pozwala na prowadzenie komunikacji w niezaufanym środowisku.
Hipoteza alternatywna HS'_2	Technologia blockchain **NIE** pozwala na prowadzenie komunikacji w niezaufanym środowisku.
Wynik weryfikacji	Hipotezę HS_2 należy przyjąć.
Uzasadnienie	Przedstawione w pracy schematy gwarantujące zachowanie integralności, poufności oraz dostępności danych przechowywanych z wykorzystaniem sieci IPFS oraz technologii blockchain takich jak Hyperledger Fabric, Secret Network i Ethereum wskazuje na możliwość prowadzenia komunikacji w środowisku sieci publicznych, które z założenia jest niezaufane.

Tabela 9 Wynik weryfikacji hipotezy szczegółowej nr 3

Hipoteza badawcza HS_3	Możliwe jest utworzenie schematu składowania danych w sieci publicznej, gwarantującego integralność i poufność treści.
Hipoteza alternatywna HS'_3	**NIE JEST** możliwe utworzenie schematu składowania danych w sieci publicznej, gwarantującego integralność i poufność treści.
Wynik weryfikacji	Hipotezę HS_3 należy przyjąć.
Uzasadnienie	Opracowany schemat uwierzytelnionego dostępu do danych przechowywanych w środowisku rozproszonym gwarantuje zachowanie integralności i poufności treści. Zarówno integralność, jak i poufność jest gwarantowana poprzez właściwe prymitywy kryptograficzne.

Tabela 10 Wynik weryfikacji hipotezy głównej

Hipoteza badawcza HG	Zastosowanie technologii blockchain pozwala na realizację uwierzytelnionego dostępu do danych zdecentralizowanych.
Hipoteza alternatywna HG'	Zastosowanie technologii blockchain **NIE** pozwala na realizację uwierzytelnionego dostępu do danych zdecentralizowanych.
Wynik weryfikacji	Hipotezę HG należy przyjąć.

Uzasadnienie	Opracowany schemat uwierzytelnionego dostępu do danych zdecentralizowanych wykorzystuje technologie sieci blockchain Ethereum i Secret Network, usługę zdecentralizowanego przechowywania danych IPFS oraz aplikację przeglądarkową uruchamianą jedynie na urządzeniu końcowym. Kombinacja powyższych technologii pozwoliła na opracowanie systemu zapewniającego poufność, integralność i dostępność przechowywanych danych. Technologia blockchain stanowiła kluczowy element rozwiązania, pozwalając na wykonywanie operacji kryptograficznych w środowisku rozproszonym, z zachowaniem poufności przechowywanych i przetwarzanych danych.

Analiza wyników weryfikacji hipotez wskazuje na przyjęcie wszystkich założonych hipotez szczegółowych oraz hipotezy głównej. Przeprowadzone analizy teoretyczne i praktyczne eksperymenty pozwalają na wyeksponowanie potencjału technologii rejestru rozproszonego, w szczególności w kontekście zapewnienia bezpieczeństwa danych i komunikacji w środowisku cechującym się ograniczonym poziomem zaufania. W rozprawie wykazano, że unikalne właściwości blockchain pozwalają na decentralizację i uwierzytelniania dostępu do danych, przy jednoczesnej eliminacji potrzeby istnienia zaufanych pośredników.

Szczególnie istotnym aspektem poruszonych badań jest identyfikacja i eksploracja nowych schematów działania, pozwalających na bezpieczne przechowywanie i wymianę informacji w scenariuszach, dla których tradycyjne rozwiązania zawodzą. Integracja technologii blockchain z nowoczesnymi prymitywami kryptograficznymi pozwala na tworzenie systemów informatycznych gotowych na wdrożenie w różnych sektorach życia społecznego i gospodarki.

Celem praktycznym pracy było zaproponowanie schematu realizacji uwierzytelnionego dostępu do danych w sposób całkowicie zdecentralizowany. Zaproponowane rozwiązania oraz implementacje stanowią fundament dla dalszych badań nad systemami zwiększającymi bezpieczeństwo i prywatność. Potwierdzenie postawionych w pracy hipotez badawczych otwiera nowe perspektywy dla rozwoju bezpiecznych i niezawodnych systemów teleinformatycznych.

Zaproponowany w rozprawie schemat uwierzytelnionego dostępu do danych ma potencjał znaczącego wpływu na praktyczne zastosowania w wielu sektorach. Możliwości integracji z systemami bankowymi, e-administracją czy systemami zarządzania infrastrukturą krytyczną otwierają nowe ścieżki dla wdrożeń cyfrowych, gdzie bezpieczeństwo danych jest

priorytetem. Dalsze badania mogą skupić się na adaptacji proponowanego schematu dla konkretnych aplikacji przemysłowych, co pozwoli na bezpośrednią demonstrację korzyści płynących z implementacji zdecentralizowanych protokołów bezpieczeństwa w kontekście rzeczywistych scenariuszy operacyjnych. **Przemyślane wdrożenie tych technologii** może również stanowić istotny element w kształtowaniu polityk ochrony danych na poziomie narodowym i międzynarodowym, **zwiększając odporność na ataki i zabezpieczając prywatność użytkowników.**

Zaprezentowane w rozprawie podejście łączy różne platformy blockchain w celu zapewnienia bezpieczeństwa i prywatności danych. Stanowi nie tylko innowację na polu technicznym, ale również otwiera drogę do szeroko zakrojonych zastosowań praktycznych. Wprowadzenie schematów umożliwiających uwierzytelnianie i decentralizację w środowisku, które tradycyjnie opiera się na centralizacji, podważa dotychczasowe metody zabezpieczeń i jest krokiem milowym w dążeniu do autonomii danych przy zachowaniu ich poufności. Tego rodzaju **przełomowe rozwiązania mogą zrewolucjonizować wiele sektorów**, **w tym e-zdrowie, obronność, sektor finansowy i administrację publiczną**, dostarczając nie tylko nowych narzędzi, ale także kształtując przyszłe **strategie bezpieczeństwa** na poziomie globalnym.

Antycypując, dysertacja ta stanowi istotny krok w kierunku zrozumienia i upowszechnienia wykorzystania potencjału blockchain do budowy systemów informatycznych cechujących się wyższym poziomem odporności, bezpieczeństwa i poszanowania prywatności użytkowników, co jest niezbędne do realizacji wyzwań stawianych przed cyfrowym społeczeństwem.

8. Bibliografia.

[1] Bayer, D., Haber, S., & Stornetta, W. S. (1993). Improving the efficiency and reliability of digital time-stamping. In *Sequences II: Methods in Communication, Security, and Computer Science*(pp. 329-334). Springer New York.

[2] Dai, W. (1998). B-money.

[3] Nakamoto S. (2008). Bitcoin: A peer-to-peer electronic cash system.

[4] Szabo, N. (2005). Bit gold. https://nakamotoinstitute.org/bit-gold.

[5] Buterin, V. (2013). Ethereum white paper. *GitHub repository*, *1*, 22-23.

[6] Popovski, L., Soussou, G., & Webb, P. B. (2018). A brief history of blockchain. *Leg. News*, 431-437.

[7] Aggarwal, S., & Kumar, N. (2021). History of blockchain-blockchain 1.0: Currency. In *Advances in Computers* (Vol. 121, pp. 147-169). Elsevier.

[8] Poon, J., & Dryja, T. (2015). The bitcoin lightning network. *Scalable o-chain instant payments*, 20-46.

[9] Armstrong, M. (2021). Ethereum, Smart Contracts and the Optimistic Roll-up.

[10] Yakovenko, A. (2018). Solana: A new architecture for a high performance blockchain v0. 8.13. *Whitepaper*.

[11] Cachin, C. (2016, July). Architecture of the hyperledger blockchain fabric. In *Workshop on distributed cryptocurrencies and consensus ledgers* (Vol. 310, No. 4, pp. 1-4).

[12] Smith, H. J. (2016). *Technical analysis of the Bitcoin cryptocurrency* (Doctoral dissertation, Hochschule für angewandte Wissenschaften Hamburg).

[13] Okupski, K. (2014). Bitcoin developer reference. *Eindhoven*, 3-4.

[14] BitcoinWiki, https://bitcoinwiki.org, dostęp 21.02.2024

[15] Nakamoto, S. (2009). *Base58.h,* dostęp 2024-02-21., https://github.com/bitcoin/bitcoin /blob/aaaaad6ac95b402fe18d019d67897ced6b316ee0/src/base58.h

[16] Chris Moore, StackExchange: How to calculate new bits value, 2013, dostęp 21.02.2024 http://bitcoin.stackexchange.com/questions/2924/how-to- calculate-new-bits-value

[17] Fu, X., Wang, H., & Shi, P. (2021). A survey of Blockchain consensus algorithms: mechanism, design and applications. *Science China Information Sciences*, *64*, 1-15.

[18] Lamport L, Shostak R, Pease M. The Byzantine generals problem. *ACM Trans Program Lang Syst*, 1982, 4: 382-401

[19] Perry, K. J., & Toueg, S. (1986). Distributed agreement in the presence of processor and communication faults. *IEEE Transactions on Software Engineering*, (3), 477-482.

[20] Lamport, L. (2019). The part-time parliament. In *Concurrency: the Works of Leslie Lamport* (pp. 277-317).

[21] Lamport, L. (2001). Paxos made simple. *ACM Sigact News*, *32*(4), 18-25.

[22] Fischer, M. J., Lynch, N. A., & Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, *32*(2), 374-382.

[23] Karame, G. O., Androulaki, E., & Capkun, S. (2012, October). Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM conference on Computer and communications security* (pp. 906-917).

[24] Karame, G. (2016, October). On the security and scalability of bitcoin's blockchain. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 1861-1862).

[25] King, S., & Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, *19*.

[26] Castro, M., & Liskov, B. (1999, February). Practical byzantine fault tolerance. In *OsDI* (Vol. 99, No. 1999, pp. 173-186).

[27] Oh, M., Ha, S., Yoon, J. H., Lee, K. W., Son, Y., & Yeom, H. Y. (2020). Graph Learning BFT: A Design of Consensus System for Distributed Ledgers. *IEEE Access*, *8*, 161739-161751.

[28] Baird, L. (2016). The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. *Swirlds Tech Reports SWIRLDS-TR-2016-01, Tech. Rep*, *34*, 9-11.

[29] Cryptocurrency's Energy Consumption Problem, https://rmi.org/cryptocurrencys-energy-consumption-problem/, dostęp 22.02.2024

[30] Eyal, I., Gencer, A. E., Sirer, E. G., & Van Renesse, R. (2016). {Bitcoin-NG}: A scalable blockchain protocol. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)* (pp. 45-59).

[31] Larimer, D. (2014). Delegated proof-of-stake (dpos). *Bitshare whitepaper*, *81*, 85.

[32] Milutinovic, M., He, W., Wu, H., & Kanwal, M. (2016, December). Proof of luck: An efficient blockchain consensus protocol. In *proceedings of the 1st Workshop on System Software for Trusted Execution* (pp. 1-6).

[33] Karantias, K., Kiayias, A., & Zindros, D. (2020). Proof-of-burn. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24* (pp. 523-540). Springer International Publishing.

[34] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., ... & Yellick, J. (2018, April). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference* (pp. 1-15).

[35] Hyperledger Fabric Simple BFT, https://jira.hyperledger.org/browse/FAB-378 dostęp 23.02.2024

[36] Buterin, V. (2013). Ethereum white paper. *GitHub repository*, *1*, 22-23.

[37] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, *151*(2014), 1-32.

[38] Jagannath, N., Barbulescu, T., Sallam, K. M., Elgendi, I., Mcgrath, B., Jamalipour, A., ... & Munasinghe, K. (2021). An on-chain analysis-based approach to predict ethereum prices. *IEEE Access*, *9*, 167972-167989.

[39] Hildenbrandt, E., Saxena, M., Zhu, X., Rodrigues, N., Daian, P., Guth, D., & Roşu, G. (2017). Kevm: A complete semantics of the ethereum virtual machine.

[40] Delmolino, K., Arnett, M., Kosba, A., Miller, A., & Shi, E. (2015). A programmer's guide to ethereum and serpent. *URL: https://mc2-umd. github. io/ethereumlab/docs/serpent_tutorial. pdf*, 22-23.

[41] Dannen, C. (2017). *Introducing Ethereum and solidity* (Vol. 1, pp. 159-160). Berkeley: Apress.

[42] Aldweesh, A., Alharby, M., Mehrnezhad, M., & Van Moorsel, A. (2019, July). OpBench: A CPU performance benchmark for Ethereum smart contract operation code. In *2019 IEEE International Conference on Blockchain (Blockchain)* (pp. 274-281). IEEE.

[43] Devroye, L. (1992). A note on the probabilistic analysis of patricia trees. *Random Structures & Algorithms*, *3*(2), 203-214.

[44] Popov, S. (2018). The tangle. *White paper*, *1*(3), 30.

[45] Guo, F., Xiao, X., Hecker, A., & Dustdar, S. (2020, December). Characterizing IOTA tangle with empirical data. In *GLOBECOM 2020-2020 IEEE Global Communications Conference* (pp. 1-6). IEEE.

[46] Silvano, W. F., & Marcelino, R. (2020). Iota Tangle: A cryptocurrency to communicate Internet-of-Things data. *Future generation computer systems*, *112*, 307-319.

[47] Li, Y., Cao, B., Peng, M., Zhang, L., Zhang, L., Feng, D., & Yu, J. (2020). Direct acyclic graph-based ledger for Internet of Things: Performance and security analysis. *IEEE/ACM Transactions on Networking*, *28*(4), 1643-1656.

[48] Blockchain vs Tangle: Untangling The IOTA Tangle, https://academy.moralis.io/blog/blockchain-vs-tangle-untangling-the-iota-tangle, dostęp 27.02.2024

[49] Popov, S., Saa, O., & Finardi, P. (2019). Equilibria in the tangle. *Computers & Industrial Engineering*, *136*, 160-172.

[50] Chohan, U. W. (2021). The double spending problem and cryptocurrencies. *Available at SSRN 3090174*.

[51] Divya, M., & Biradar, N. B. (2018). IOTA-next generation block chain. *Int. J. Eng. Comput. Sci*, *7*(04), 23823-23826.

[52] The Tangle: an Illustrated Introduction, https://blog.iota.org/the-tangle-an-illustrated-introduction-4d5eae6fe8d4/, dostęp 27.02.2024

[53] Geyer, C. J. (1992). Practical markov chain monte carlo. *Statistical science*, 473-483.

[54] Popov, S., Moog, H., Camargo, D., Capossele, A., Dimitrov, V., Gal, A., ... & Attias, V. (2020). The coordicide. *Accessed Jan*, 1-30.

[55] Suhail, S., Hussain, R., Khan, A., & Hong, C. S. (2020). Orchestrating product provenance story: When IOTA ecosystem meets electronics supply chain space. *Computers in Industry*, *123*, 103334.

[56] Gangwani, P., Perez-Pons, A., Bhardwaj, T., Upadhyay, H., Joshi, S., & Lagos, L. (2021). Securing environmental IoT data using masked authentication messaging protocol in a DAG-based blockchain: IOTA tangle. *Future Internet*, *13*(12), 312.

[57] Drzewo skrótów, https://commons.wikimedia.org/wiki/File:Hash_Tree-pl.svg, dostęp 27.02.2024

[58] Becker, G. (2008). Merkle signature schemes, merkle trees and their cryptanalysis. *Ruhr-University Bochum, Tech. Rep*, *12*, 19

[59] Brogan, J., Baskaran, I., & Ramachandran, N. (2018). Authenticating health activity data using distributed ledger technologies. *Computational and structural biotechnology journal*, *16*, 257-266.

[60] Zheng, X., Sun, S., Mukkamala, R. R., Vatrapu, R., & Ordieres-Meré, J. (2019). Accelerating health data sharing: A solution based on the internet of things and distributed ledger technologies. *Journal of medical Internet research*, *21*(6), e13583.

[61] Iota: Signature and Validation, https://medium.com/@abmushi/iota-signature-and-validation-b95b3f9ec534, dostęp 28.02.2024

[62] Handy, P. (2017). Introducing masked authenticated messaging. *IOTA Foundation Medium blog*.

[63] Shafeeq, S., Zeadally, S., Alam, M., & Khan, A. (2019). Curbing address reuse in the iota distributed ledger: A cuckoo-filter-based approach. *IEEE Transactions on Engineering Management*, *67*(4), 1244-1255.

[64] Buchmann, J., Dahmen, E., Ereth, S., Hülsing, A., & Rückert, M. (2011). On the security of the Winternitz one-time signature scheme. In *Progress in Cryptology–AFRICACRYPT 2011: 4th International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011. Proceedings 4* (pp. 363-378). Springer Berlin Heidelberg.

[65] Buchmann, J., & Ding, J. (Eds.). (2008). *Post-Quantum Cryptography: Second International Workshop, PQCrypto 2008 Cincinnati, OH, USA October 17-19, 2008 Proceedings* (Vol. 5299). Springer Science & Business Media.

[66] Rohde, S., Eisenbarth, T., Dahmen, E., Buchmann, J., & Paar, C. (2008). Fast hash-based signatures on constrained devices. In *Smart Card Research and Advanced Applications: 8th IFIP WG 8.8/11.2 International Conference, CARDIS 2008, London, UK, September 8-11, 2008. Proceedings 8* (pp. 104-117). Springer Berlin Heidelberg.

[67] Sarfraz, U., Alam, M., Zeadally, S., & Khan, A. (2019). Privacy aware IOTA ledger: Decentralized mixing and unlinkable IOTA transactions. *Computer Networks*, *148*, 361-372.

[68] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., ... & Muralidharan, S. (2018, April). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference* (pp. 1-15).

[69] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. ACM Comput. Surv., 22(4):299–319, 1990.

[70] B. Kemme and G. Alonso. A new approach to developing and implementing eager database replication protocols. ACM Transactions on Database Systems, 25(3):333–379, 2000.

[71] C.Cachin,R.Guerraoui,andL.E.T.Rodrigues.IntroductiontoReliableandSecure Distributed Programming (2. ed.). Springer, 2011.

[72] A. N. Bessani, J. Sousa, and E. A. P. Alchieri. State machine replication for the masses with BFT-SMART. In International Conference on Dependable Systems and Networks (DSN), pages 355–362, 2014.

[73] J. Sousa, A. Bessani, and M. Vukolić. A Byzantine fault-tolerant ordering service for the Hyperledger Fabric blockchain platform. In International Conference on Dependable Systems and Networks (DSN), 2018.

[74] Benet, J. (2014). Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*.

[75] Williams, S., Diordiiev, V., Berman, L., & Uemlianin, I. (2019). Arweave: A protocol for economically sustainable information permanence. *arweave.org, Tech. Rep*.

[76] Bieri, C. (2021). An overview into the InterPlanetary File System (IPFS): use cases, advantages, and drawbacks. *Communication Systems XIV*, *28*.

[77] Martino, R., & Cilardo, A. (2020). SHA-2 acceleration meeting the needs of emerging applications: A comparative survey. *IEEE Access*, *8*, 28415-28436.

[78] Aumasson, J. P., Meier, W., Phan, R. C. W., & Henzen, L. (2014). The hash function BLAKE.

[79] TURNER, S. (2010). Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms draft-turner-md5-seccon-update-06. txt. *http://tools. ietf. org/html/draft-turner-md5-seccon-update-06*.

[80] Bleichenbacher, D., & Maurer, U. M. (1994, August). Directed acyclic graphs, one-way functions and digital signatures. In *Annual International Cryptology Conference* (pp. 75-82). Berlin, Heidelberg: Springer Berlin Heidelberg.

[81] InterPlanetary File System: Merkle Directed Acyclic Graphs (DAGs); https:// docs.ipfs.io/concepts/merkle-dag, dostęp 29.02.2024

[82] InterPlanetary File System: Directed acyclic graphs (DAGs); https:// docs.ipfs.io/concepts/how-ipfs-works/#directed-acyclic-graphs-dags, dostęp 29.02.2024

[83] De la Rocha, A., Dias, D., & Psaras, Y. (2021). Accelerating content routing with bitswap: A multi-path file transfer protocol in ipfs and filecoin. *San Francisco, CA, USA*, 11.

[84] Psaras, Y., & Dias, D. (2020, June). The interplanetary file system and the filecoin network. In *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)* (pp. 80-80). IEEE.

[85] InterPlanetary File System: Distributed hash tables (DHTs); https : / / docs.ipfs.io/concepts/how-ipfs-works/#distributed-hash-tables-dhts

[86] Starnberger, G., Kruegel, C., & Kirda, E. (2008, September). Overbot: a botnet protocol based on Kademlia. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks* (pp. 1-9).

[87] Freedman, M. J., Freudenthal, E., & Mazieres, D. (2004, March). Democratizing content publication with coral. In *NSDI* (Vol. 4, pp. 18-18).

[88] Baumgart, I., & Mies, S. (2007, December). S/kademlia: A practicable approach towards secure key-based routing. In *2007 International conference on parallel and distributed systems* (pp. 1-8). IEEE.

[89] Maymounkov, P., & Mazieres, D. (2002, March). Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems* (pp. 53-65). Berlin, Heidelberg: Springer Berlin Heidelberg.

[90] Bitswap, https://docs.ipfs.tech/concepts/bitswap/, dostęp 01.03.2024

[91] Bauer, D. P. (2022). Filecoin. In *Getting Started with Ethereum: A Step-by-Step Guide to Becoming a Blockchain Developer* (pp. 97-101). Berkeley, CA: Apress.

[92] Guidi, B., Michienzi, A., & Ricci, L. (2021, December). A libP2P Implementation of the Bitcoin Block Exchange Protocol. In *Proceedings of the 2nd International Workshop on Distributed Infrastructure for Common Good* (pp. 1-4).

[93] Agostinho, P., Dias, D., & Veiga, L. (2022, September). Smartpubsub: Content-based pub-sub on ipfs. In *2022 IEEE 47th Conference on Local Computer Networks (LCN)* (pp. 327-330). IEEE.

[94] libp2p: NAT Traversal; https://docs.libp2p.io/concepts/nat, dostęp 01.03.2024

[95] Thaler, D., & Aboba, B. (2008). *What makes for a successful protocol?* (No. rfc5218).

[96] Williams, S., Kedia, A., Berman, L., & Campos-Groth, S. (2023). Arweave: The Permanent Information Storage Protocol.

[97] Williams, S., Diordiiev, V., Berman, L., & Uemlianin, I. (2019). Arweave: A protocol for economically sustainable information permanence. *arweave. org, Tech. Rep*.

[98] Ralph Merkle. “Secrecy, authentication, and public key systems”. PhD thesis. 1979. url: https://link.springer.com/content/pdf/10.1007/0-387- 34805-0_21.pdf.

[99] Lamriji, Y., Kasri, M., El Makkaoui, K., & Beni-Hssane, A. (2023, May). A comparative study of consensus algorithms for blockchain. In *2023 3rd International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)* (pp. 1-8). IEEE.

[100] Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (2018). *Handbook of applied cryptography*. CRC press.

[101] Boneh, D., Bonneau, J., Bünz, B., & Fisch, B. (2018, July). Verifiable delay functions. In *Annual international cryptology conference* (pp. 757-788). Cham: Springer International Publishing.

[102] Yakovenko, A. (2018). Solana: A new architecture for a high performance blockchain v0. 8.13. *Whitepaper*.

[103] Kowalski, L. (2001). *Statystyka*. Wydział Cybernetyki-Wojskowa Akademia Techniczna.

[104] Guidi, B., Michienzi, A., & Ricci, L. (2022, November). Evaluating the decentralisation of filecoin. In *Proceedings of the 3rd International Workshop on Distributed Infrastructure for the Common Good* (pp. 13-18).

[105] de Figueiredo, S., Madhusudan, A., Reniers, V., Nikova, S., & Preneel, B. (2021, March). Exploring the storj network: A security analysis. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing* (pp. 257-264).

[106] Vorick, D., & Champine, L. (2014). Sia: Simple decentralized storage., https://blockchainlab.com/pdf/whitepaper3.pdf, dostęp 11.03.2024.

[107] Schueffel, P. (2021). DeFi: Decentralized finance-an introduction and overview. *Journal of Innovation Management*, *9*(3), I-XI.

[108] El Faqir, Y., Arroyo, J., & Hassan, S. (2020, August). An overview of decentralized autonomous organizations on the blockchain. In *Proceedings of the 16th international symposium on open collaboration* (pp. 1-8).

[109] Hammi, B., Zeadally, S., & Perez, A. J. (2023). Non-fungible tokens: a review. *IEEE Internet of Things Magazine*, *6*(1), 46-50.

[110] Bauer, D. P. (2022). ERC-20: Fungible Tokens. In *Getting Started with Ethereum: A Step-by-Step Guide to Becoming a Blockchain Developer* (pp. 17-48). Berkeley, CA: Apress.

[111] Liquidity in the DeFi sector recovering in 2023, https://en.cryptonomist.ch/2023/06/10/liquidity-defi-sector-recovering-2023/, dostęp 12.03.2024

[112] Lo, Y. C., & Medda, F. (2021). Uniswap and the Emergence of the Decentralized Exchange. *Journal of financial market infrastructures*, *10*(2), 1-25.

[113] Xu, J., & Vadgama, N. (2022). From banks to defi: the evolution of the lending market. *Enabling the Internet of Value: How Blockchain Connects Global Businesses*, 53-66.

[114] Soska, K., Dong, J. D., Khodaverdian, A., Zetlin-Jones, A., Routledge, B., & Christin, N. (2021, April). Towards understanding cryptocurrency derivatives: A case study of BitMEX. In *Proceedings of the Web Conference 2021* (pp. 45-57).

[115] Popescu, A. D. (2020). Transitions and concepts within decentralized finance (Defi) Space. *Research Terminals in the social sciences*.

[116] Bartoletti, M., Chiang, J. H. Y., & Lluch-Lafuente, A. (2022). A theory of automated market makers in defi. *Logical Methods in Computer Science*, *18*.

[117] Hassan, S., & De Filippi, P. (2021). Decentralized autonomous organization. *Internet Policy Review*, *10*(2), 1-10.

[118] RobotEra Docs, https://docs.robotera.io, dostęp 12.03.2024

[119] Calvaria: Duels of Eternity White Paper, https://whitepaper.calvaria.io/decentralized-autonomous-organization/about-dao, dostęp 12.03.2024

[120] Battle Infinity Whitepaper, https://whitepaper.battleinfinity.io/welcome-to-battle-infinity-whitepaper-2.0/, dostęp 12.03.2024

[121] Kim, J. (2021). Regulation of Decentralized Systems: A Study of Uniswap. *Harv. JL & Tech.*, *35*, 335.

[122] Sushiswap, https://www.sushi.com, dostęp 12.03.2024

[123] ERC-721: Non-Fungible Token Standard, https://eips.ethereum.org/EIPS/eip-721, dostęp 12.03.2024

[124] Hammi, B., Zeadally, S., & Perez, A. J. (2023). Non-fungible tokens: a review. *IEEE Internet of Things Magazine*, *6*(1), 46-50.

[125] M. Glet, K. Kaczyński, NFT-BASED User Authentication Scheme For Mobile APPS, 39th IBIMA Conference, 2022

[126] Berners-Lee, T. (1998). Uniform resource identifiers (URI): Generic syntax and semantics. *RFC 2396*.

[127] Valeonti, F., Bikakis, A., Terras, M., Speed, C., Hudson-Smith, A., & Chalkias, K. (2021). Crypto collectibles, museum funding and OpenGLAM: challenges, opportunities and the potential of Non-Fungible Tokens (NFTs). *Applied Sciences*, *11*(21), 9931.

[128] ERC-1155, https://eips.ethereum.org/EIPS/eip-1155, dostęp 12.03.2024

[129] Munoz, M. F., Zhang, K., & Amara, F. (2022, May). ZipZap: A blockchain solution for local energy trading. In *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (pp. 1-5). IEEE.

[130] Vitelaru, E., & Persia, L. (2023). Fractional vehicle ownership and revenue generation through blockchain asset tokenization. *Transport and Telecommunication*, *24*(2), 120-127.

[131] Comparing Etehereum Token Standards, https://tradedog.io/comparing-ethereum-token-standards-erc1155-vs-erc721/, dostęp 12.03.2024

[132] Stallings, W., & Grażyński, A. (2012). *Kryptografia i bezpieczeństwo sieci komputerowych: matematyka szyfrów i techniki kryptologii*. Helion.

[133] Turner, J. M. (2008). The keyed-hash message authentication code (hmac). *Federal Information Processing Standards Publication*, *198*(1), 1-13.

[134] Kaja, D. V. S., Fatima, Y., & Mailewa, A. B. (2022). Data integrity attacks in cloud computing: A review of identifying and protecting techniques. *J. homepage www. ijrpr. com ISSN*, *2582*, 7421.

[135] DLT: Blockchain DAG and Tempo, https://sapcrmtutorial.blogspot.com/p/blockchain.html, dostęp 13.03.2024

[136] Shrier, D., Wu, W., & Pentland, A. (2016). Blockchain & infrastructure (identity, data security). *Massachusetts Institute of Technology-Connection Science*, *1*(3), 1-19.

[137] Džaferović, E., Sokol, A., Abd Almisreb, A., & Norzeli, S. M. (2019). DoS and DDoS vulnerability of IoT: A review. *Sustainable Engineering and Innovation*, *1*(1), 43-48.

[138] Jiang, Q., Lee, Y. C., & Zomaya, A. Y. (2020). The limit of horizontal scaling in public clouds. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, *5*(1), 1-22.

[139] Sy, M. P. M., Marasigan, R. I., & Festijo, E. D. (2023, June). Hyperledger-operated blockchain integration: writing, deploying and testing custom chaincode. In *2023 Sixth International Symposium on Computer, Consumer and Control (IS3C)* (pp. 151-154). IEEE.

[140] Zhai, Z., Shen, S., & Mao, Y. (2023). A Toolbox for Migrating the Blockchain-Based Application From Ethereum to Hyperledger Fabric. *The Computer Journal*, bxad061.

[141] Gaur, N., O'Dowd, A., Novotny, P., Desrosiers, L., Ramakrishna, V., & Baset, S. A. (2020). *Blockchain with hyperledger fabric: Build decentralized applications using hyperledger fabric 2*. Packt Publishing Ltd.

[142] Androutsellis-Theotokis, S., & Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM computing surveys (CSUR)*, *36*(4), 335-371.

[143] Abdullah Lajam, O., & Ahmed Helmy, T. (2021, February). Performance evaluation of IPFS in private networks. In *2021 4th International Conference on Data Storage and Data Engineering*(pp. 77-84).

[144] 45 Global DDoS Attack Statistics 2024, https://www.getastra.com/blog/security-audit/ddos-attack-statistics/, dostęp 14.03.2024

[145] Hupperich, T. (2023, October). On DDoS Attacks as an Expression of Digital Protest in the Russo-Ukrainian War 2022. In *2023 International Symposium on Networks, Computers and Communications (ISNCC)* (pp. 1-8). IEEE.

[146] Mansfield-Devine, S. (2011). DDoS: threats and mitigation. *Network Security*, *2011*(12), 5-12.

[147] Trautwein, D., Raman, A., Tyson, G., Castro, I., Scott, W., Schubotz, M., ... & Psaras, Y. (2022, August). Design and evaluation of IPFS: a storage layer for the decentralized web. In *Proceedings of the ACM SIGCOMM 2022 Conference* (pp. 739-752).

[148] Sridhar, S., Ascigil, O., Keizer, N., Genon, F., Pierre, S., Psaras, Y., ... & Król, M. (2023). Content Censorship in the InterPlanetary File System. *arXiv preprint arXiv:2307.12212*.

[149] Doan, T. V., Psaras, Y., Ott, J., & Bajpai, V. (2022). Toward decentralized cloud storage with IPFS: opportunities, challenges, and future considerations. *IEEE Internet Computing*, *26*(6), 7-15.

[150] Balduf, L., Henningsen, S., Florian, M., Rust, S., & Scheuermann, B. (2022, July). Monitoring data requests in decentralized data storage systems: A case study of IPFS. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)* (pp. 658-668). IEEE.

[151] Ko, H., Oh, J., & Kim, S. U. (2023). Digital Content Management Using Non-Fungible Tokens and the Interplanetary File System. *Applied Sciences*, *14*(1), 315.

[152] Fotiou, N., Siris, V. A., & Polyzos, G. C. (2021, June). Enabling self-verifiable mutable content items in IPFS using Decentralized Identifiers. In *2021 IFIP Networking Conference (IFIP Networking)* (pp. 1-6). IEEE.

[153] Waldo, J., Wyant, G., Wollrath, A., & Kendall, S. (1996, July). A note on distributed computing. In *International Workshop on Mobile Object Systems* (pp. 49-64). Berlin, Heidelberg: Springer Berlin Heidelberg.

[154] Wang, J., Jiang, C., Han, Z., Ren, Y., Maunder, R. G., & Hanzo, L. (2017). Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones. *Ieee vehIcular technology magazIne*, *12*(3), 73-82.

[155] Gregory, D. (2011). From a view to a kill: Drones and late modern war. *Theory, culture & society*, *28*(7-8), 188-215.

[156] Spezio, A. E. (2002). Electronic warfare systems. *IEEE Transactions on Microwave Theory and Techniques*, *50*(3), 633-644.

[157] Cullen, T. M. (2011). *The MQ-9 Reaper remotely piloted aircraft: Humans and machines in action* (Doctoral dissertation, Massachusetts Institute of Technology).

[158] Dalsjö, R., Jonsson, M., & Norberg, J. (2023). A brutal examination: Russian military capability in light of the Ukraine war. In *Survival: June-July 2022* (pp. 7-28). Routledge.

[159] Tianfeng, F., Xiaojing, M., & Chi, Z. (2023, June). Development status of anti UAV swarm and analysis of new defense system. In *Journal of Physics: Conference Series* (Vol. 2478, No. 9, p. 092011). IOP Publishing.

[160] Sichitiu, M. L. (2005, May). Wireless mesh networks: opportunities and challenges. In *Proceedings of World Wireless Congress* (Vol. 2, p. 21).

[161] Basaez Serey, J. (2023). HL-DRIP: A Blockchain-based Remote Drone ID Protocol registry management: Evaluation of a Hyperledger Fabric-based solution to manage DRIP registries.

[162] Ongaro, D., & Ousterhout, J. (2015). The raft consensus algorithm. *Lecture Notes CS*, *190*, 2022.

[163] Yang, G., Lee, K., Lee, K., Yoo, Y., Lee, H., & Yoo, C. (2022). Resource analysis of blockchain consensus algorithms in hyperledger fabric. *IEEE Access*, *10*, 74902-74920.

[164] Achakri, M. (2021). A Survey on Anonymization Techniques for Blockchain Transactions. *Communication Systems XIV*, 134.

[165] Costan, V., & Devadas, S. (2016). Intel SGX explained. *Cryptology ePrint Archive*.

[166] Woetzel, C. (2022). Secret Network: A Privacy-Preserving Secret Contract & Decentralized Application Platform., https://www.securesecrets.org/Secret_Network_Graypaper_2.0.1_1.pdf, dostęp 18.03.2024

[167] Mo, F., Shamsabadi, A. S., Katevas, K., Demetriou, S., Leontiadis, I., Cavallaro, A., & Haddadi, H. (2020, June). Darknetz: towards model privacy at the edge using trusted execution environments. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services* (pp. 161-174).

[168] Mofrad, M. H., & Lee, A. (2017). Leveraging Intel SGX to create a nondisclosure cryptographic library. *arXiv preprint arXiv:1705.04706*.

[169] Wang, X., Sheng, P., Kannan, S., Nayak, K., & Viswanath, P. (2022). Trustboost: Boosting trust among interoperable blockchains. *Cryptology ePrint Archive*.

[170] Essaid, M., Kim, J., & Ju, H. (2023). Inter-Blockchain Communication Message Relay Time Measurement and Analysis in Cosmos. *Applied Sciences*, *13*(20), 11135.

[171] Privacy Model of Secret Contracts, https://build.scrt.network/dev/privacy-model-of-secret-contracts.html, dostęp 18.03.2024

[172] Zyskind, G., Encryption specification: Contracts State Encryption, https://github.com/scrtlabs/SecretNetwork/blob/e0ed66f/docs/protocol/encryption-specs.md#contracts-state-encryption, dostęp 18.03.2024

[173] Krawczyk, H., & Eronen, P. (2010). *HMAC-based extract-and-expand key derivation function (HKDF)* (No. rfc5869).

[174] Zyskind, G., Encryption specification: Theoretical Attacks, https://github.com/enigmampc/SecretNetwork/blob/e0ed66f/docs/protocol/encryption-specs.md#Theoretical-Attacks, dostęp 18.03.2024

[175] Green, M., & Ateniese, G. (2007). Identity-based proxy re-encryption. In *Applied Cryptography and Network Security: 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007. Proceedings 5* (pp. 288-306). Springer Berlin Heidelberg.

[176] Bentaleb, A., Zhan, Z., Tashtarian, F., Lim, M., Harous, S., Timmerer, C., ... & Zimmermann, R. (2022, October). Low latency live streaming implementation in DASH and HLS. In *Proceedings of the 30th ACM International Conference on Multimedia* (pp. 7343-7346).

[177] Pote, S., Sule, V., & Lande, B. K. (2019, April). Arithmetic of Koblitz Curve Secp256k1 Used in Bitcoin Cryptocurrency Based on One Variable Polynomial Division. In *2nd International Conference on Advances in Science & Technology (ICAST)*.

[178] Bertoni, G., Daemen, J., Peeters, M., & Van Assche, G. (2013, May). Keccak. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 313-314). Berlin, Heidelberg: Springer Berlin Heidelberg.

[179] Gayoso Martínez, V., Hernández Encinas, L., & Sánchez Ávila, C. (2010). A survey of the elliptic curve integrated encryption scheme.

[180] Halpin, H. (2014, April). The W3C web cryptography API: motivation and overview. In *Proceedings of the 23rd International Conference on World Wide Web* (pp. 959-964).

[181] Elliptic Github repository, https://github.com/indutny/elliptic, dostęp 20.03.2024

[182] Eth-crypto Github repository, https://github.com/pubkey/eth-crypto, dostęp 20.03.2024

[183] Js-ipfs Github repository, https://github.com/ipfs/js-ipfs, dostęp 20.03.2024

[184] Secret.js Github repository, https://github.com/scrtlabs/secret.js, dostęp 20.03.2024

[185] Barker, E. (2020). Recommendation for Key Management: Part 1—General (NIST Special Publication 800-57 Part 1 Revision 5). *National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA*.

[186] McGrew, D., & Viega, J. (2004). The Galois/counter mode of operation (GCM). *submission to NIST Modes of Operation Process*, *20*, 0278-0070.

[187] Deirmentzoglou, E., Papakyriakopoulos, G., & Patsakis, C. (2019). A survey on long-range attacks for proof of stake protocols. *IEEE access*, *7*, 28712-28725.

[188] Cryptography, P. K. Elliptic Curves in Public Key Cryptography: The Diffie Hellman Key Exchange Protocol and its relationship to the Elliptic Curve Discrete Logarithm Problem Public Key Cryptography.

[189] Bernstein, D. J. (2006). Curve25519: new Diffie-Hellman speed records. In *Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9* (pp. 207-228). Springer Berlin Heidelberg.

[190] van Schaik, S., Seto, A., Yurek, T., Batori, A., AlBassam, B., Garman, C., ... & Yarom, Y. (2022). Sok: Sgx. fail: How stuff get exposed., https://sgx.fail, dostęp 21.03.2024

[191] Notice: Successful Resolution of xAPIC Vulnerability on Secret Network, https://scrt.network/blog/notice-successful-resolution-of-xapic-vulnerability/, dostęp 21.03.2024

[192] Gawinecki, J. A., & Szmidt, J. (2002). *Zastosowanie ciał skończonych i krzywych eliptycznych w kryptografii*. Wojskowa Akademia Techniczna. Instytut Matematyki i Badań Operacyjnych. Wydział Cybernetyki.

9. Spis tabel

10. Spis rysunków