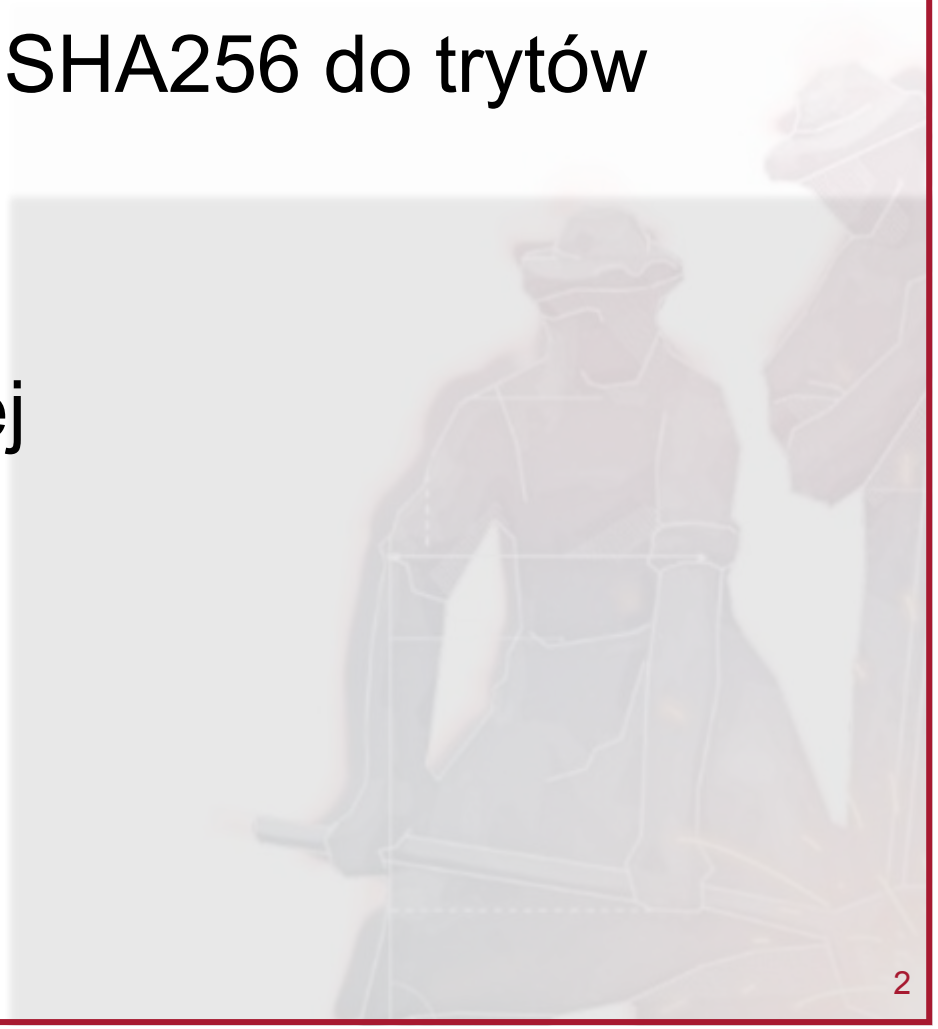
Analiza i adaptacja kryptograficzna algorytmów haszujących w kontekście systemów ternarnych

Autor: Mieszko Makowski
Opiekun naukowy: dr hab. inż., prof AGH **Marek Natkaniec**

Teleinformatyki i Cyberbezpieczeństwa

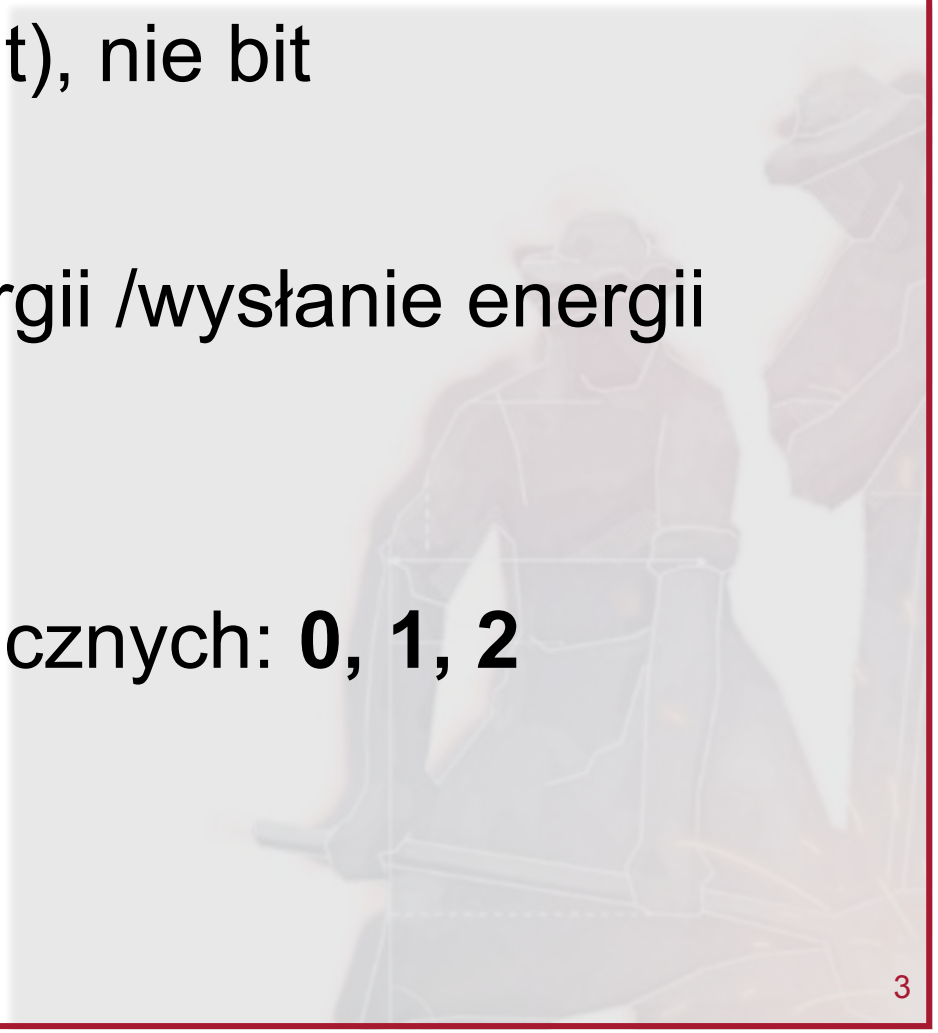
Cele projektu

- Przystosowanie algorytmów SHA-1 i SHA256 do trytów
- Implementacja operacji logicznych
- Porównanie wersji binarnej i ternarnej



Czym jest system ternarny ?

- Każda jednostka informacji to **tryt** (trit), nie bit
- Budowa na podstawie: pobranie energii /wysłanie energii /brak działanie
- System oparty na trzech stanach logicznych: **0, 1, 2**



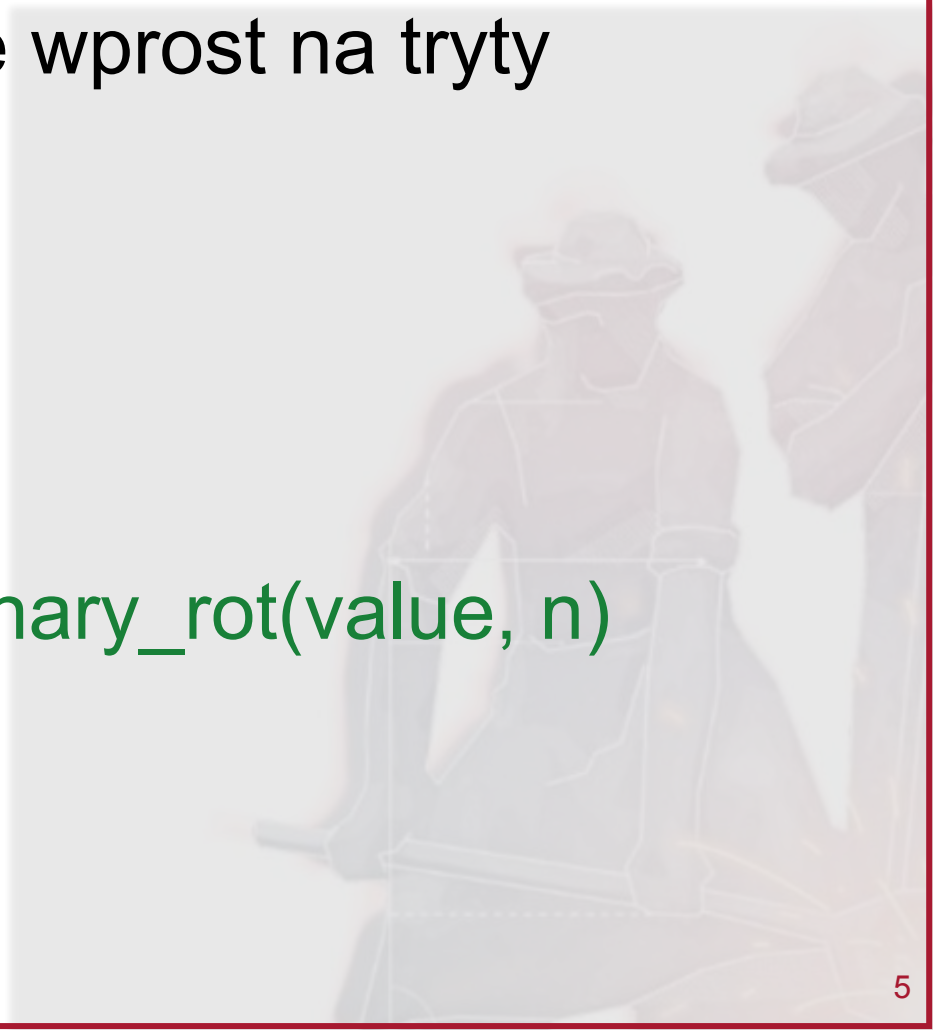
Algorytmy haszujące

- Operacje ściśle zależne od arytmetyki binarnej
- **SHA-1**: 160-bitowy skrót, 80 rund, 5 rejestrów
- **SHA-256**: 256-bitowy skrót, 64 rundy, 8 rejestrów
- Kluczowe elementy: padding wiadomości, podział na bloki, funkcje rund

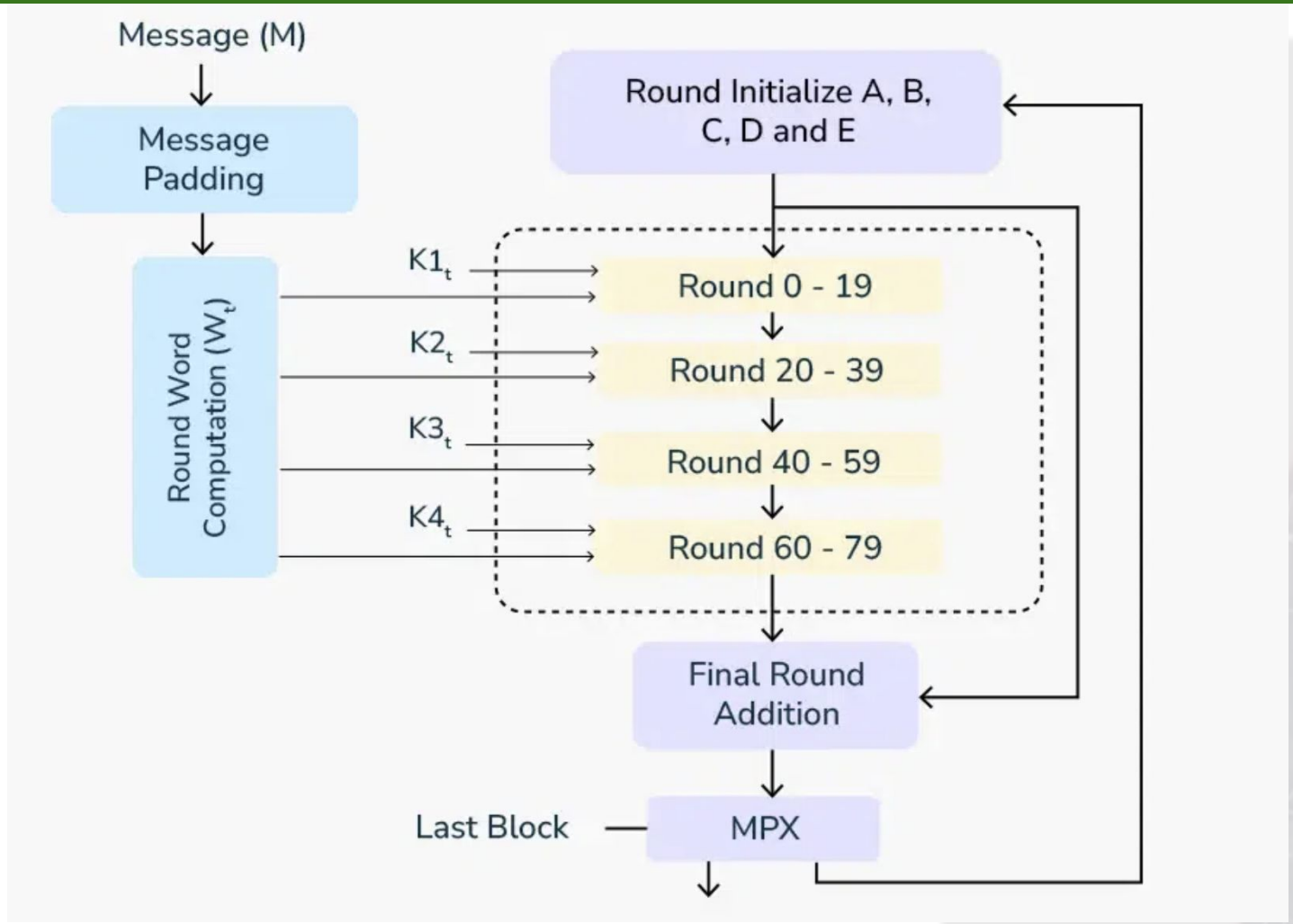
Przekształcenie operacji binarnych

- Operacje na bitach nie przenoszą się wprost na tryty
- Tylko operacje komutatywne
- Zaimplementowano:

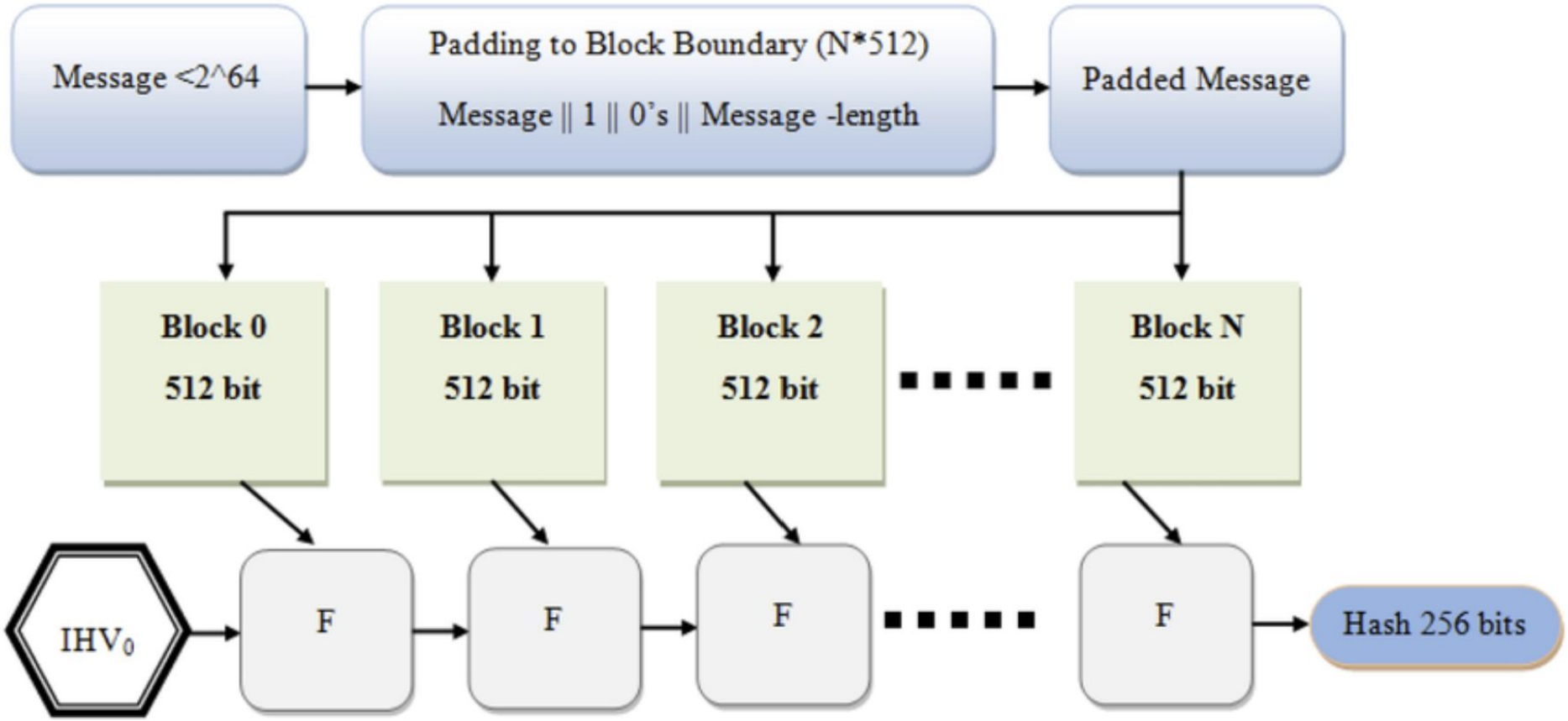
ternary_xor(a, b), ternary_and(a, b), ternary_rot(value, n)



SHA-1


Message (M)
Message Padding
Round Word Computation (W_t)
Round Initialize A, B, C, D and E
$K1_t$
$K2_t$
$K3_t$
$K4_t$
Round 0 - 19
Round 20 - 39
Round 40 - 59
Round 60 - 79
Final Round Addition
Last Block
MPX

SHA256


Message <2^64
Padding to Block Boundary (N*512)
Message || 1 || 0's || Message -length
Padded Message
Block 0
512 bit
Block 1
512 bit
Block 2
512 bit
Block N
512 bit
IHV0
F
F
F
F
Hash 256 bits

Operacje binarne vs Operację ternarne

Wejściowe	00/01/11	00/01/02/11/12/22
Wyjściowe	0/1	0/1/2
Liczba przekształceń	8	728
Przekształcenia Użyteczne	XOR, AND, OR, NOT XOR, NOT AND, NOT OR,	725 użytecznych przekształceń komutatywnych

AND

wybieranie mniejszej wartości z dwóch

Przekształcenie binarne	
00	0
01	0
11	1

Przekształcenie tenarne	
00	0
01	0
02	0
11	1
12	1
22	2

OR

wybieranie większej wartości z dwóch

Przekształcenie binarne	
00	0
01	1
11	1

Przekształcenie tenarne	
00	0
01	1
02	2
11	1
12	2
22	2

XOR

dodawanie modulo (mod 2 / mod 3)

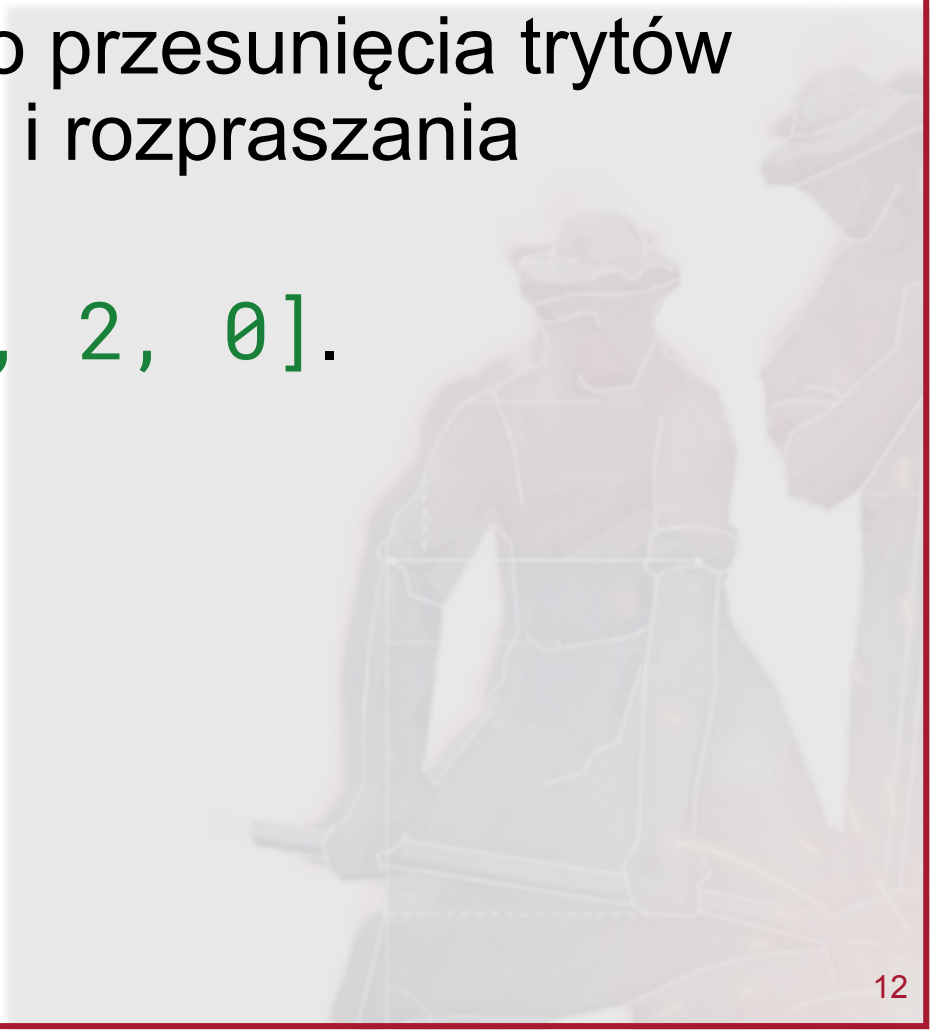
Przekształcenie binarne	
00	0
01	1
11	0

Przekształcenie tenarne	
00	0
01	1
02	2
11	2
12	0
22	1

ROT

- ROT (rotacja) to operacja cyklicznego przesunięcia trytów w prawo lub lewo służy do mieszania i rozpraszania danych

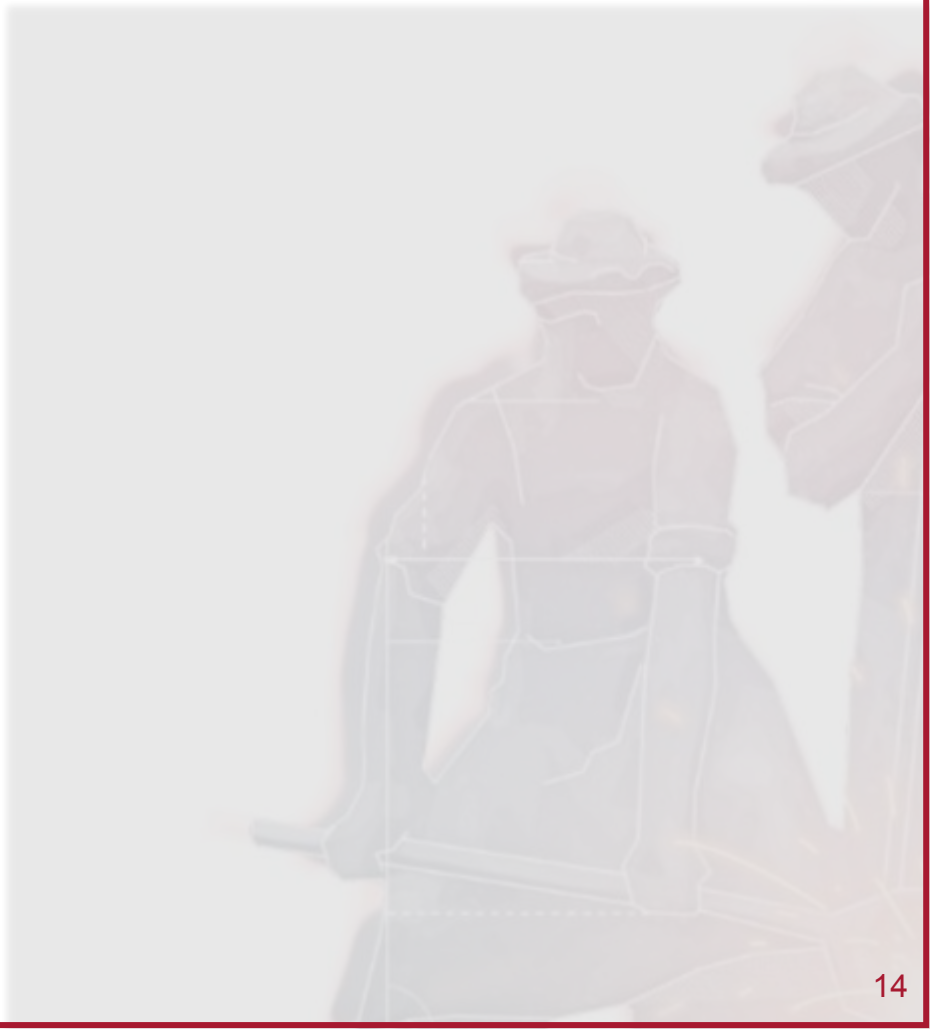
Przykład: `ROT([0, 1, 2], 1)` → `[1, 2, 0]`.



Jak przystosowałem SHA ?

- Padding zmieniony na obsługę trytową
- Dane wejściowe zamienione z bitów na tryty
- Rejestry i funkcje rund przemapowane na logikę ternarną
- Zmiana operacji matematycznej na wersję trójwartościową

Porównanie budowy SHA



SHA-1 binarne vs SHA-1 tenarne

Element	Binarne	Ternarne
Rejestry	5 x 32 bity	5 x 21 trytów
Rundy	80	80
Operacje	ROTL, XOR	ROT trytów, ternary_xor
Dane wejściowe	512 bit	342 tryty

SHA-256 binarne vs SHA-256 tenarne

Element	Binarne	Ternarne
Rejestry	8 x 32 bity	8 x 21 trytów
Rundy	64	64
Operacje	ROTR, SHR	ternary_rot, ternary_shr
Dane wejściowe	512 bit	342 tryty

Zastosowanie narzędzi w Pythonie

- Zdefiniowałem własny typ danych Tryt, obsługujący wartości 0–1–2
- Zaimplementowałem funkcje logiczne: ternary_xor(), ternary_and(), ternary_or()
- Napisałem konwertery bitów do trytów i odwrotnie
- Funkcje rot, shr, add zostały dostosowane do operacji trójwartościowych

Przeprowadzenie analizy

- Generowałem skróty dla tych samych danych w wersji binarnej i ternarnej
- Sprawdzałem jak zmiana jednego bitu/trytu wpływa na wynik (dyfuzja)
- Zliczałem kolizje (czy różne dane dają taki sam skrót)
- Obliczałem entropię wyjścia — jak równomiernie rozkładają się wartości
- Mierzyłem czas obliczeń funkcji SHA-1 i SHA-256 w obu wersjach

SHA-1 binarne vs SHA-1 tenarne

Cecha	Binarne	Ternarne
Skrót	160 bit	105 trytów
Dyfuzja	dobra	wyższa entropia
Kolizyjność	średnia	porównywalna
Wydajność	wysoka	niższa

SHA-256 binarne vs SHA-256 tenarne

Cecha	Binarne	Ternarne
Skrót	256 bit	168 trytów
Operacje	zoptymalizowane	wolniejsze
Złożoność	niska	wyższa logicznie
Bezpieczeństwo	bardzo dobre	potencjalnie lepsze (entropia)

Podsumowanie

- SHA-1 łatwiejszy do adaptacji niż SHA-256
- Tryty wymagają całkowicie nowych operacji
- Balanced ternary zwiększa entropię i symetrię
- Wersja ternarna potencjalnie bezpieczniejsza, ale mniej wydajna
- Python pozwolił na szybką prototypową implementację

Dziękuję za uwagę