QubitCoin v2.0 - (30-40)

- QubitCoin

QubitCoin

2025 12 6

RubikPoW QubitCoin QBC QubitCoin

1

QubitCoin QBC RubikPoW PoW
QubitCoin RubikPoW

2

2.1

RSA ECC

2.2

-
-
  RSA ECDSA

2.3

NIST

1.
2.
3.
4.

3 RubikPoW

3.1

n×n×n G_n

3.1 ( ). *n×n×n*

$$|G_n| = \frac{8! \cdot 3^7 \cdot 12! \cdot 2^{11} \cdot \prod_{i=1}^{\lfloor (n-2)/2 \rfloor} (24!)^i}{2} \cdot \frac{24!^{\lfloor (n-3)/2 \rfloor}}{2}$$

.

- 8 3 7
- 12 2 11
- $\lfloor (n-2)/2 \rfloor$ 24
-

n=3 $|G_3| = 43,252,003,274,489,856,000 \approx 4.3 \times 10^{19}$
n=4 $|G_4| \approx 7.4 \times 10^{45}$
n=5 $|G_5| \approx 2.8 \times 10^{74}$ □

3.2

n×n×n NP

3.3

RubikPoW

n×n×n

$$T_{classical} = O(|G_n|)$$

$$T_{quantum} = O(\sqrt{|G_n|})$$

n=3

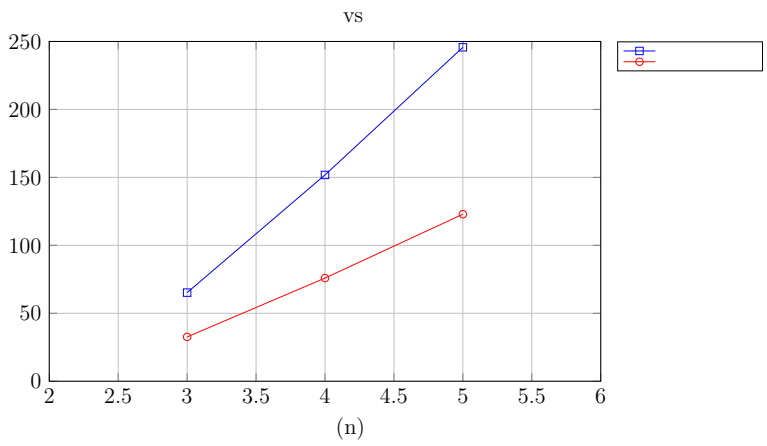
$$T_{classical} \approx 2^{65.2}, \quad T_{quantum} \approx 2^{32.6}$$

n=4

$$T_{classical} \approx 2^{151.8}, \quad T_{quantum} \approx 2^{75.9}$$

n=5

$$T_{classical} \approx 2^{245.7}, \quad T_{quantum} \approx 2^{122.9}$$



1:

3.4

RubikPoW O(k) k

RubikPoW

1.

2.

3. $i = 0$ 7

   - $state.corners[i].position \neq i$ OR $state.corners[i].orientation \neq 0$
   - **return** False

4. $i = 0$ 11

- $state.edges[i].position \neq i$ OR $state.edges[i].orientation \neq 0$
- **return** False

5. $i = 0$ $NumCenters(state.size)$
   - $state.centers[i].position \neq i$
   - **return** False
6. **return** True

4 RubikPoW

4.1

QubitCoin

```
struct RubikBlock {
    uint32 version;
    bytes32 prev_block_hash;
    bytes32 merkle_root;
    uint32 timestamp;
    uint32 difficulty;                    //      n
    uint8 cube_size;                      // n×n×n n
    uint16 max_moves_allowed;             //
    bytes32 initial_cube_state;          //
    bytes32 final_cube_state;            //
    uint16 solution_length;              //
    uint8[solution_length] solution;     //
    uint64 nonce;                        //
    bytes32 block_hash;                  //
    Transaction[] transactions;          //
}
```

4.2

1.
2. A* IDA*
3.
4.
5.

4.3

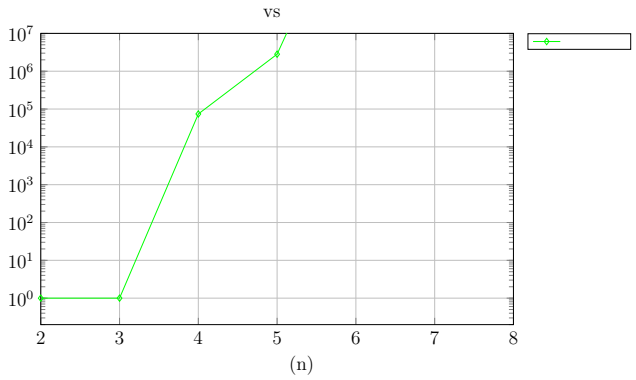
RubikPoW

- n×n×n n
-
-

$$D_{total} = D_{size}(n) \cdot D_{moves}(k) \cdot D_{hash}(target)$$

where:

$$D_{size}(n) = \log_2(|G_n|)/\log_2(|G_3|) \quad (1)$$
$$D_{moves}(k) = \quad (2)$$
$$D_{hash}(target) = 2^{256}/target \quad (3)$$



2:

5

5.1 PoW

	Shor	**Grover**		
SHA-256	N/A	$2^{128} \rightarrow 2^{64}$		
Scrypt	N/A	$2^{128} \rightarrow 2^{64}$		
Equihash Zcash	N/A	$2^{n/2} \rightarrow 2^{n/4}$		
RSA-2048	2^{112}	N/A		
ECC-P256	2^{128}	N/A	DLP	
RubikPoW-n	N/A	$\sqrt{\lvert G_n \rvert}$		

1:

5.2

RubikPoW

1. IDA*
2.
3.
4.

5.3

RubikPoW

6

6.1

	(QBC)	**%**
	21,000,000	100%
PoW	14,700,000	70%
	4,200,000	20%
	2,100,000	10%

2: QubitCoin

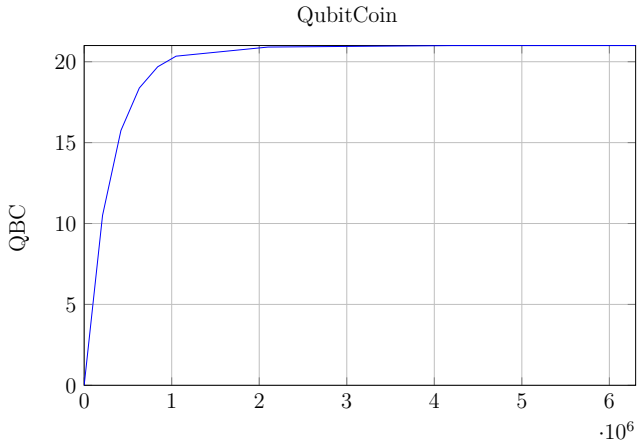
6.2

QubitCoin RubikPoW

- 210,000 4
- 1 50QBC
- 2140
- 2,100

6.3

- 40%
- 25%
- 20%
- 15%



3: QubitCoin

7

7.1 2025-2026

2025 4	v1.0	
2026 1		
2026 2		QubitCoin
2026 3	SDK	SDK
2026 4	DEX	

7.2 2027-2029

2027 1		
2027 2		
2027 3		Layer-2
2027 4		
2028 1		
2028 2	DApps	
2029 4		

8

8.1

QubitCoin Substrate

- RubikPoW
- RubikPoW
- Peer-to-Peer Libp2p

-

8.2 RubikPoW

RubikPoW

```
pub struct Pallet<T>(PhantomData<T>);

impl<T: Config> Pallet<T> {
    pub fn submit_solution(
        origin,
        solution: Vec<Move>,
        nonce: u64
    ) -> DispatchResult {
        //
        ensure_signed(origin)?;

        //
        Self::validate_solution(&solution)?;

        //
        Self::check_difficulty(&solution, nonce)?;

        //
        Self::process_reward(&sender)?;

        Ok(())
    }

    fn validate_solution(solution: &[Move]) -> bool {
        //
        let mut state = Self::get_initial_state();
        for move in solution {
            state.apply_move(move);
        }

        //
        state.is_solved()
    }

    fn check_difficulty(solution: &[Move], nonce: u64) -> bool {
        let hash = Self::calculate_block_hash(solution, nonce);
        hash < Self::get_current_target()
    }
}
```

8.3

```
pub struct RubiksCubeState {
    corners: [CornerPiece; 8],
    edges: [EdgePiece; 12],
    centers: Vec<CenterPiece>,
    n: u8,  //     : n×n×n
}

#[derive(Copy, Clone, PartialEq)]
pub enum CornerPiece {
    Solved(u8),      //
    Permuted(u8, u8) //
}

#[derive(Copy, Clone, PartialEq)]
pub enum EdgePiece {
    Solved(u8),
    Permuted(u8, u8)
}

pub enum Move {
    U, Up, U2,        //
    D, Dp, D2,        //
    L, Lp, L2,        //
    R, Rp, R2,        //
    F, Fp, F2,        //
    B, Bp, B2,        //
    //
    Uw, Dm,  ...    //
}
```

9

9.1

QubitCoin 7-10 10 Layer-2

9.2

RubikPoW PoW

9.3

10

10.1

1.

	(USD)	Watts/Tx	(kg)
	$0.25	1520	0.08
	$1.50	45	0.015
QubitCoin ( )	$0.15	85	0.04

5:

2.

3.

4. RubikPoW

10.2

- SDK Rust, JavaScript, Python
- RESTful API
-
-

11

11.1

-
-
-
-

11.2

1. **51%** PoW
2.
3.
4.
5.

12

12.1 DeFi

QubitCoin DeFi

-
-
-

12.2

-
-
-

12.3

-
- -proof
-

13 RubikPoW

13.1

n×n×n G_n

13.1 ( ). G_n k *RubikPoW*

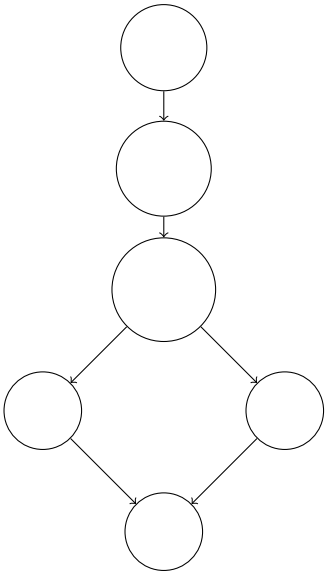
$$\rho(n,k) = \frac{N_{solutions}(n,k)}{|G_n|} \approx \frac{12^k}{|G_n|} \cdot f(n)$$

$f(n)$

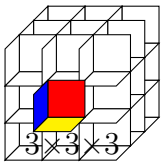
13.2

$s_1, s_2 \in G_n$

$$d_H(s_1, s_2) = \sum_{i=1}^{N_{pieces}} \delta(p_i(s_1), p_i(s_2))$$



4: RubikPoW



5: 3×3×3

14

[1] Shor, P.W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 124-134.

[2] Grover, L.K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 212-219.

15

QubitCoin
RubikPoW
QubitCoin 30 40 QubitCoin
QubitCoin

16